UNIVERSITE BRETAGNE LOIRE

MINES Nantes
Institut Mines-Télécom

Thèse de Doctorat

Axel GRIMAULT

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'École nationale supérieure des Mines de Nantes
sous le sceau de l'Université Bretagne Loire

École doctorale : Sciences et Technologies de l'Information, et Mathématiques (STIM)

Discipline : Informatique et applications, section CNU 27
Spécialité : Informatique et recherche opérationnelle
Unité de recherche : Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN)

Soutenue le 16 juin 2016
Thèse n° : 2016EMNA0222

Optimisation de tournées de camions complets dans le secteur des travaux publics

JURY

Rapporteurs :	**M. André LANGEVIN**, Professeur associé, École Polytechnique de Montréal **M. Éric SANLAVILLE**, Professeur, Université du Havre
Examinateurs :	**M. Pierre DEJAX**, Professeur - Chargé de mission, École des Mines de Nantes **M. Jorge MENDOZA**, Maître de conférences, École Polytechnique de l'Université de Tours **M. Marc SEVAUX**, Professeur, Université de Bretagne Sud
Invité :	**M. Willy LAMBERT**, Directeur des systèmes d'information, Luc Durand
Directrice de thèse :	**M^me^ Nathalie BOSTEL**, Professeur, Université de Nantes
Co-encadrant de thèse :	**M. Fabien LEHUÉDÉ**, Maître assistant HDR, École des Mines de Nantes

Remerciements

Au commencement il n'y avait rien.

Amélie Nothomb
Métaphysique des tubes

Si je peux aujourd'hui écrire ces lignes, c'est par le fruit d'un grand nombre de rencontres inédites et riches. Il y a quelques années, je n'imaginais pas parcourir ce chemin mais, derrière moi, il y a surement eu la métaphysique qui m'y a poussé. À mesure que j'avance, je découvre la richesse et l'importance des échanges humains et précieux qui m'ont nourri. Ici, je souhaite remercier toutes celles et ceux qui m'ont accompagné. Milles excuses à celles et ceux que j'oublierai.

Je souhaite remercier, en premier lieu, Nathalie et Fabien. Il y a quatre années, vous m'avez fait confiance en me permettant de réaliser cette thèse et me donnant l'opportunité de découvrir le monde de la recherche et de l'enseignement. Durant toutes ces années, nous avons eu beaucoup d'échanges passionnés sur un sujet que nous nous sommes accaparés au fil des années. Et des années, il y en a eues ! Beaucoup trop... il fallait finir. Vous les avez vues défiler mais vous m'avez toujours soutenu dans les extrema locaux de ma motivation. Plus que des conseils, vous avez été accompagnateurs dans mes démarches et m'avez transmis rigueur et pertinence dans la production scientifique. J'ai aimé découvrir et apprendre à vos côtés ce métier et je vous en remercie infiniment. La seule entorse à cette rigueur sera d'employer le mot *route* dans ces remerciements et non *tournée*.

Je remercie également, chaleureusement, les membres du jury d'avoir accepté de juger mon travail : les rapporteurs, André Langevin et Éric Sanlaville, pour la qualité des retours que vous m'avez faits ainsi que les examinateurs, Pierre Dejax, Jorge Mendoza et Marc Sevaux pour nos échanges lors de la soutenance. Nos routes se sont croisées et j'espère qu'elles se recroiseront de nouveaux.

Cette thèse s'est inscrite dans le cadre du projet ORLoGES. Jusqu'au dernier jour, j'aurais buté sur cet acronyme. Mais derrière, restent des personnes, en particulier celles de l'entreprise Luc Durand. Merci Willy pour toutes les discussions techniques et les décisions que nous avons eues lors de ce projet. Tu m'as aussi fait découvrir de nombreuses technologies et je t'ai fait (re)découvrir le merveilleux univers de la fanfare (surtout à ton fils). Nicolas, anciennement collègue de couloir, qui blanchit du code maintenant. Merci à Jean-Christophe Louvet d'avoir guidé ce projet tout au long de ces années et de croire en la digitalisation des entreprises des travaux publics. En règle générale, vous étiez tous bien accompagnés par Marylène, Michel, Anthony. Mes amitiés aux autres partenaires du projet et surtout aux plus proches Thomas et Gwenaëlle de Greenspector.

A l'Université de Nantes, mes premiers pas en recherche opérationnelle ont été guidés par ce grand belge Xavier et ce grand coureur Anthony. C'est un peu grâce à vous que j'appartiens à la même famille. Vous m'avez fait découvrir la RO, l'informatique, le monde universitaire. Ce master fut une superbe expérience et m'a motivé à continuer tant que je le pouvais dans cette voie. Une pensée pour tous mes compagnons de master, en particulier Salim.

42 semaines à peu près. Vous ne saviez pas si j'allais vraiment finir ce marathon. Et des marathons j'en ai fait, 4 précisément. Celui-ci fut de loin le plus long en temps. Ils ont été mes confidents, mes guides éclairés et éclairants, des compagnons de fortune, des acolytes de courses à pied, un hommage à mes collègues du bureau C025b : Philippe, Juliette, Thomas (Yo !), Yuan et Tianyu. Tous les autres doctorants du départements, les actuel·le·s (Alex, Houda, Jiuchin, Johan, Ka Yu, Lori, Oscar, Quentin, Steven, Sylvain, Xiao, Yulong), les ancien·ne·s (Agnès, Carlos, Clément, Jérémy, Majid, Mi, Renaud, Tanguy, Thomas, Yier) et les post-doctorant·e·s (Alan, Hoang, Wenjin). Mes remerciements à tous les collègues du département (anciens et nouveaux) qui m'ont accompagnés dans les épreuves scientifiques, humaines et sportives. Une pensée très amicale aux secrétaires du département (Anita et Isabelle) qui de loin nous regardent grandir. Mes amitiés très fortes à toute la communauté chinoise pour le partage de culture que nous avons expérimenté. Enfin, à tous les doctorants, je vous souhaite d'arriver à ce moment où il ne reste qu'extase. Une pensée éternelle pour vous.

Je réalise maintenant, à l'écriture de ces mots et avec émotion, tout le bonheur d'avoir été entouré par des personnes avec lesquelles il est important de partager passions et intimité. Mes colocataires de La Marrière, Émilie, Killian, Mira, Laurent, Nicolas, Florent, Keven, Eva, Melania, Naira. Vous avez été d'une grande richesse ces dernières années. J'espère continuer de partager avec vous tous les plaisirs de la vie. Enfin, que serait ma vie sans ma deuxième famille, mes amis musiciens de l'Orchestre de Cuivres et Percussions, de l'Amfifanfare, de la Fanfare du Coin et de la Grande Pièce et tous ceux avec lesquels j'ai pu joué de la trompette pour plaisir et bonheur. Merci pour tous les concerts, voyages et vacations musicales où les rencontres furent un grand bol d'air.

Pour finir, les plus proches, Michèle et Bernard, mes parents, qui m'ont toujours laissé choisir ma voie. Le même choix que vous laissez à Lambert et Chloé, mon frère et ma sœur, qui m'accompagnent sur les sentiers et sur les portées. Merci à cette famille extraordinaire, je souhaite qu'il nous reste beaucoup de moments de bonheur et de joie ensemble.

Enfin, il m'a fallu beaucoup d'énergie pour finir ce manuscrit. Tout mon amour va pour celle qui m'a accompagné dans les derniers moments de ma thèse. Avec plaisir, je continuerai de partager notre amour sur la route qui nous attend.

ZIVELI !

Table des matières

1

Introduction

Ce chapitre présente le cadre global de réalisation de cette thèse. Dans la section 1.1, nous présentons l'ancrage de ce problème dans le secteur industriel. Cette thèse a été réalisée dans le cadre d'un projet collaboratif de recherche appliquée dont les grandes lignes sont présentées dans la section 1.2. Nous présentons les enjeux scientifiques auxquels cette thèse répond dans la section 1.3. Enfin, le plan de la thèse est présenté dans la section 1.4.

1.1 Présentation du problème

Le travail présenté dans cette thèse se positionne dans la thématique des transports de marchandises pour le secteur des travaux publics. En 2013, plus de 50% des activités de travaux publics ont été réalisées pour des opérations de travaux routiers et de terrassement représentant un chiffre d'affaires de 23 milliards d'euros [46]. Pour réaliser ces opérations, des véhicules transportent des marchandises d'une origine vers une destination tout en interagissant avec l'environnement dans lequel ils évoluent. Nous nous intéressons au transport des matériaux nécessaires à la construction des infrastructures routières, au terrassement, à la déconstruction, à la voirie et aux aménagements. Ces transports s'effectuent majoritairement avec des véhicules terrestres, pour la plupart des camions. Cette activité de transport est communément nommée Transport Routier de Marchandises (TRM).

Le transport routier de voyageurs ou de marchandises est un secteur concurrentiel et cohabite avec d'autres solutions de transport notamment le ferroviaire, le fluvial et l'aérien. En dépit de la conjoncture économique défavorable dans la fin de la première décennie du XXIème siècle, les immatriculations des véhicules neufs sont reparties à la hausse. Cependant, nous pouvons observer une nette diminution des transports. Pour cela, nous nous basons sur l'indicateur *tonnes.km* qui indique la quantité en tonnes de marchandises transportées par la distance parcourue. Pour les véhicules, sous pavillon français, de type poids lourds (PTAC > 3.5 tonnes), le TRM confirme une tendance à la baisse entre les années 2007 et 2014 (voir figure 1.2) [74].

Force est de constater, aujourd'hui, la prédominance des impacts environnementaux liés aux émissions des Gaz à Effet de Serre (GES) dans le débat public. Environ 27% des émissions de GES sont liées aux activités de transport (tout secteur confondu). De plus, les transports peuvent engendrer d'autres impacts, dits sociétaux, tels que le bruit, la détérioration des chaussées, l'accidentologie dans les zones urbaines. Souvent présentés en second plan, ces effets secondaires représentent un véritable coût financier mais également humain. Mieux maîtriser les déplacements est donc un enjeu environnemental et sociétal.

Même si les études [74] divergent sur l'exactitude des chiffres annoncés, nous remarquons la contri-

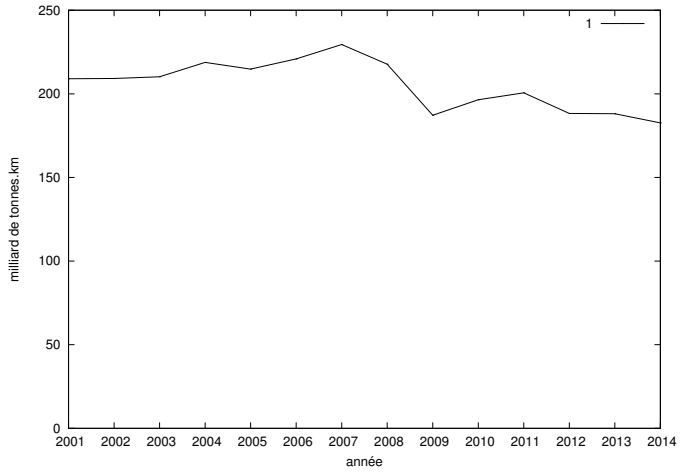


FIGURE 1.1 – Évolution du transport routier de marchandises sous pavillon français entre les années 2000 et 2014.

bution majeure des TRM sur les émissions de GES. Fort heureusement, des actions de politique publique viennent apporter des solutions à ces problématiques. Ces actions se basent souvent sur des études scientifiques qui viennent justifier et étayer la portée des actions. Une des disciplines à la frontière entre les mathématiques et l'informatique, la recherche opérationnelle, contribue à l'élaboration de ces études. La recherche opérationnelle propose des outils d'aide à la décision pour répondre à des problématiques industrielles, notamment celle des transports. Afin d'optimiser les déplacements en proposant le trajet le plus court, Traveling Salesman Problem (TSP) résolu par [29], des algorithmes d'optimisation permettent de résoudre, aujourd'hui, les problèmes liés au transport de marchandises et/ou de personnes. Au cours des dernières décennies, nous observons une plus forte prise en compte des enjeux environnementaux et sociétaux dans la littérature scientifique et industrielle.

La gestion centralisée des opérations logistiques offre la possibilité de prendre en compte un nombre plus important de paramètres, tels que la géolocalisation des véhicules ou bien le trafic du réseau routier, pour l'élaboration de tournées de véhicules. De l'informatisation des ordres de transport à l'information en temps réel, en passant par le transport multi-modal, il existe de plus en plus de leviers pour permettre une planification juste et optimisée de ces tournées. Le secteur des travaux publics réinvente aujourd'hui les anciens codes du métier d'hier. Enjeu économique indéniable, mais aussi objectif sociétal, la gestion efficiente d'une flotte de camions et de chauffeurs est devenue un véritable métier de logisticien auquel la recherche opérationnelle offre des outils d'aide à la décision.

Cette thèse se positionne dans le secteur du Transport Routier de Marchandises et expose une étude scientifique proposant des outils d'optimisation pour une utilisation industrielle dans une entreprise de travaux publics, Luc Durand.

1.2 Description du projet industriel

Cette thèse s'inscrit dans le cadre du projet industriel ORLoGES[1], partenariat entre différentes institutions publiques et privées.

Ce projet a pour objectif de proposer une suite logicielle d'aide à la gestion logistique des transports liés aux entreprises de travaux publics. Les décisions concernées sont associées à la planification et l'optimisation des transports. Les aspects sociétaux, sécuritaires et liés au développement durable font partie des enjeux et de l'innovation de ce projet. Les données liées aux flux de matériaux des chantiers existants

[1] **O**ptimisation du **R**éseau **Lo**gistique en **G**énie civil, avec les aspects **É**conomiques et **S**ociétaux / Programme d'Investissements d'Avenir

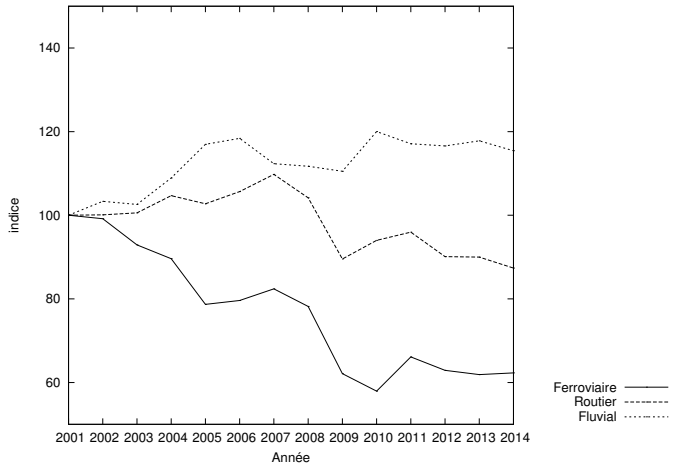


FIGURE 1.2 – Évolution du transport ferroviaire, routier et maritime en France depuis 2000 en considérant l'année 2000 comme indice 100.

et en prévision sont enregistrées et analysées pour proposer une solution qui permet de maîtriser la chaîne logistique de l'entreprise, plus particulièrement :

- optimisation des lieux d'implantation des plateformes de stockage des matériaux ;
- module de gestion des chantiers ;
- gestion de la flotte de véhicules ;
- suivi en temps réel des véhicules ;
- traçabilité des matériaux ;
- mesure de l'impact de la circulation.

Le projet *ORLoGES* est réalisé par un consortium d'entreprises et une institution publique qui sont :

- Luc Durand[2] : entreprise privée de travaux publics. Elle est à l'initiative du projet et porteuse du projet. Ses activités de travaux publics l'amènent sans cesse à innover dans la démarche qualité. Elle porte le projet de développer un outil multiservice de gestion d'activités pour une entreprise de travaux publics.
- École des Mines de Nantes[3] : école d'ingénieurs généralistes. C'est au travers de l'équipe de recherche SLP (Systèmes Logistiques et de Production) rattachée au Laboratoire IRCCyN[4] (Institut de Recherche en Communications et Cybernétique de Nantes - UMR 6597) que l'École des Mines de Nantes apporte sa compétence et son expertise de la chaîne logistique. Les enseignants-chercheurs de cette équipe SLP ont développé une forte compétence sur l'optimisation de la logistique.
- PTV Loxane[5] : entreprise privée d'édition de logiciels et Système d'Information Géographique (SIG). PTV Loxane propose des logiciels pour calculer des itinéraires et les évaluer en tenant compte de différentes normes. Leur moteur d'optimisation permet de connaître précisément la distance, le temps ainsi que les émissions en GES entre deux points.

[2]http ://www.durandtp.fr
[3]http ://www.mines-nantes.fr
[4]http ://www.irccyn.ec-nantes.fr
[5]http ://www.ptvgroup.com

- Kaliterre[6] : entreprise privée spécialisée dans les systèmes d'information. Elle valorise le développement des nouvelles technologies d'information en s'intéressant à leurs consommations énergétiques. Elle apporte son savoir faire *Développement Durable* sur les problématiques environnementales et sociétales.

- Opal Conseil : entreprise privée de conseil en organisation de la stratégie logistique. Elle a pour but d'accompagner l'ensemble des partenaires dans la structuration de la chaîne logistique.

Ce projet est soutenu par le Programme d'Investissements d'Avenir (PIA) et labellisé par le pôle de compétitivité **NOV@LOG**[7].

Dans la suite de ce document, nous nommerons l'entreprise de travaux publics Luc Durand par *l'entreprise*.

En vue de proposer une solution pour maîtriser la chaîne logistique, le projet *ORLoGES* a été découpé en plusieurs lots, chacun répondant à une problématique précise. Dans les champs liés à l'optimisation et à la recherche opérationnelle, nous nous intéressons plus précisément aux trois lots suivants :

- lot 4 (planification stratégique) : aide à la décision sur la conception du réseau logistique ;

- lot 5 (planification tactique) : aide à la décision sur la planification et l'ordonnancement des ressources aux plateformes ;

- lot 6 (planification opérationnelle) : aide à la décision sur l'optimisation des transports.

Les lots mentionnés ci-dessus sont en interaction. Chaque lot correspond à une décision logistique et les lots interagissent entre eux. Ainsi, les données de sortie d'un lot sont transformées en données d'entrée pour le lot suivant. Nous détaillons pour chacun de ces lots, les données d'entrée et de sortie ainsi que la fonctionnalité du lot.

Le lot 4 concerne la conception d'un réseau logistique dans lequel il faut décider de l'implémentation de plateformes de stockage. L'objectif de ce réseau est de minimiser :

- les coûts d'infrastructure ;

- les coûts de transports d'approvisionnement en matériaux ;

- les coûts de transports pour la distribution des matériaux aux chantiers.

Le lot 4 est un problème d'optimisation. L'objectif est de déterminer la localisation des plateformes de stockage des matériaux, la localisation des sites d'enfouissement de matériaux et le choix des modes de transport des matériaux entre chaque chantier et plateforme. Cette optimisation au niveau stratégique permet de prendre des décisions d'implantations de plateformes sur un horizon de temps annuel.

Les données d'entrée du lot 4 sont les suivantes :

- localisation des chantiers, des carrières, des plateformes existantes et des sites d'enfouissement ;

- localisation potentielle pour les futurs sites ;

- caractéristique de chaque site et de chaque chantier (durée, demande en matériaux) ;

- matrice de coût, temps et distance du réseau considéré.

Après le lancement de l'optimisation, les sorties du lot 4 sont les suivantes :

[6] http ://greenspector.com
[7] http ://www.novalog.eu

- emplacements des sites retenus pour les plateformes de stockage ;
- affectation des chantiers aux plateformes et sites ;
- mode de transport des matériaux pour chaque arc du réseau.

Le lot 5 apporte une réponse tactique à la gestion des ressources aux plateformes. Nous introduisons la notion de ressource définie comme le service qu'offre un équipement de la chaîne logistique de l'entreprise. Certaines plateformes de matériaux de l'entreprise sont critiques pour l'entreprise car elles ne peuvent accueillir qu'un nombre limité de véhicules à la fois pour des opérations de chargement ou de déchargement. Pour cela, il est nécessaire de planifier les chantiers (gestion des stocks, approvisionnement en matériaux), les flux de véhicules sur ces plateformes en respectant les contraintes de capacité. L'objectif de ce lot 5 est de fournir une solution avec une granularité temporelle de l'ordre de la semaine.

Les données d'entrée du lot 5 sont les suivantes :

- localisation des plateformes ;
- disponibilité des ressources sur les différents sites et plateformes (lot 4) ;
- demande des chantiers, planification des travaux de chaque chantier ;
- matrice de coût, temps et distance du réseau considéré.

Après le lancement de l'optimisation, les sorties du lot 5 sont les suivantes :

- affectation des transports aux ressources sur une semaine ou plus ;
- affectation des véhicules aux demandes de transport à la semaine ou plus.

Enfin, le lot 6 correspond aux décisions opérationnelles du projet *ORLoGES* telles que la gestion des déplacements des véhicules et l'ordonnancement sur les ressources. Les lots 4 et 5 établissent, lors de leur exécution, un réseau précis ainsi qu'une répartition sur les différents sites des demandes en matériaux provenant des chantiers de l'entreprise. En se basant sur ces éléments, le lot 6 peut proposer une solution pour le transport des matériaux depuis les lieux de stockage ou d'extraction vers les chantiers ainsi que l'enlèvement des matériaux des chantiers vers les lieux de stockage et d'enfouissement.

Le problème d'optimisation associé consiste à définir les tournées des véhicules de l'entreprise sur une période tout en minimisant la distance parcourue, le temps de transport et le nombre de véhicules utilisés. La gestion des transports évolue de manière dynamique en fonction des aléas (trafic, climat, chantier ajourné, etc.) et implique la planification de deux façons différentes. La première planification est la production d'un plan de transport optimisé en fonction de la connaissance des informations fournies. La seconde planification envisagée est la considération dynamique, en temps réel, des ordres de transport. Les planifications retournées par le lot 6 valent pour une journée de travail ou bien une semaine avec chaque journée détaillée, en fonction de la granularité de résolution souhaitée par le décideur.

Les données d'entrée du lot 6 sont les suivantes :

- disponibilité des ressources (lot 5) ;
- quantité de matériaux à fournir pour chaque chantier ;
- données sur les véhicules et les chauffeurs ;
- matrice de coût, temps et distance du réseau considéré.

Après le lancement de l'optimisation, les sorties du lot 6 sont les suivantes :

- planning de tournées pour les véhicules sur un horizon temporel défini ;
- affectation des véhicules aux demandes de transport à la semaine ou plus.

La figure 1.3 représente l'organigramme des lots avec leurs interactions. Les lots 4 et 5 du projet *ORLoGES* ont été réalisés indépendamment de cette thèse par des chercheurs postdoctoraux. Les travaux dans cette thèse concernent uniquement le lot 6. Plus précisément, nous étudierons le problème statique d'optimisation des tournées sur une journée ou une semaine. La prise en compte des aléas pouvant survenir au cours de l'exploitation ne fait pas partie du cadre d'étude de cette thèse.

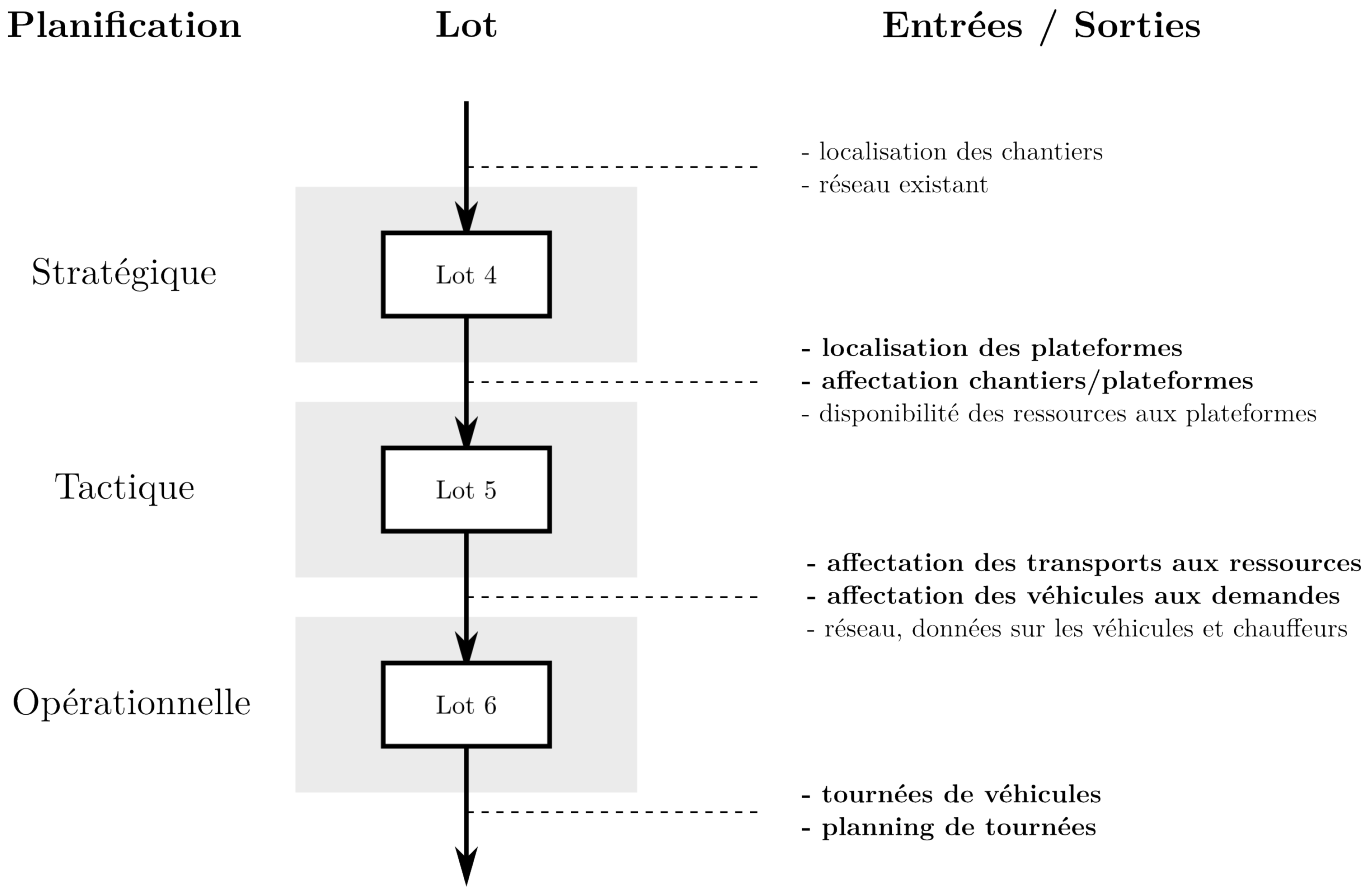


FIGURE 1.3 – Organigramme des lots du projet *ORLoGES*. Les éléments en gras représentent les sorties d'un lot qui sont utilisées pour le lot suivant.

1.3 Objectifs scientifiques

Cette thèse apporte une réponse à la problématique des transports dans le cadre du projet industriel *ORLoGES*. L'optimisation des transports est largement abordée dans la littérature scientifique de la discipline de la recherche opérationnelle et est référencée par la dénomination *tournées de véhicules* (Vehicle Routing Problem (VRP)). Cette thèse répond à deux problématiques majeures : (i) les transports en camions complets et (ii) la synchronisation de tournées sur des ressources. Nous présentons en détail les verrous scientifiques associés dans les sections suivantes.

1.3.1 Transports en camions complets

Le transport en camions complets, ou en camions pleins, désigne le transport de marchandises ou matériaux lorsqu'un camion transporte uniquement un seul type de produits au maximum de sa capacité. Le transport

en camions complets est présenté dans la littérature sous le nom de Full Truckloads [33]. Ce terme désigne un problème lorsque des véhicules, généralement des camions, doivent effectuer plusieurs trajets dans un réseau pour satisfaire des demandes et chacun de ces trajets est effectué avec un chargement complet.

Nous différencions deux cas : (i) le flotte des camions réalisant ces trajets est homogène ou (ii) la flotte des camions est hétérogène. En effet, la décision n'est pas la même selon les deux types de flotte possible. Lorsque la flotte des camions est homogène, tous les camions possèdent la même capacité de transport. Ainsi, le nombre de trajets nécessaires pour servir une demande de quantité d par des camions de capacité $c \mid c < d$ est $\lceil \frac{d}{c} \rceil$. Ce nombre de trajets est facile à établir et la décision se porte uniquement sur le nombre de camions devant effectuer les trajets (un ou plusieurs camions).

En revanche, il existe des problèmes où la décision est plus difficile à prendre lorsque la flotte de camions est hétérogène et celle ci contient au moins deux capacités différentes de camions. Pour servir cette même demande de quantité d avec des camions de capacité $c_1, c_2 \mid c_1 \neq c_2 \wedge c_1 < d \wedge c_2 < d$, plusieurs solutions sont envisageables :

- $\lceil \frac{d}{c_1} \rceil$ trajets de camions de capacité c_1 ;
- $\lceil \frac{d}{c_2} \rceil$ trajets de camions de capacité c_2 ;
- n_1 trajets de camions de capacité c_1 et n_2 trajets de camions de capacité n_2 de telle sorte que $n_1 * c_1 + n_2 * c_2 \geq d$.

La décision ne porte plus sur le nombre de camions à choisir mais sur le nombre de trajets à affecter à chaque type de camion. Pour des demandes de quantité inférieure à la capacité des véhicules où chaque véhicule peut visiter plusieurs fois une demande, le problème de tournées de véhicules associé est dénommé Split Delivery Vehicle Routing Problem (SDVRP) [36, 6, 3]. Pour des problèmes où la quantité de demande est supérieure à la capacité des véhicules, plusieurs dénominations existent dans la littérature (*full loads* ou *full truckloads*) ; celle que nous utiliserons dans le problème étudié est dite "en camions complets".

Le coût d'une solution d'un problème de tournées de véhicules est le coût défini par chacune des tournées des camions. Ce coût est calculé lorsque chaque tournée est établie, ce qui suppose que chaque partie de demande ait été affectée à un camion. L'ordonnancement des demandes sur chaque tournée de camion peut différer d'une solution à l'autre.

L'affectation optimale des demandes aux camions est difficile à déterminer. Ce problème constitue un des premiers verrous scientifiques auquel nous apportons une réponse.

1.3.2 Synchronisation aux ressources

Les problèmes de tournées de véhicules ont été résolus depuis plusieurs décennies avec différentes variantes. Dans le contexte actuel d'optimisation de la logistique, les acteurs du marché cherchent de nouvelles façons pour rendre la chaîne logistique plus efficiente. Cela se traduit par une plus forte interdépendance des flux et des tournées des véhicules notamment via des hubs logistiques ou des plateformes de mutualisation.

Cette interdépendance est modélisée dans les problèmes de tournées de véhicules par une contrainte dite de synchronisation. Dans un état de l'art [35], Drexl identifie quatre types de contraintes de synchronisation : (i) de tâches, (ii) d'opérations, (iii) de chargement et (iv) de ressources. Dans notre problème, nous nous intéresserons aux synchronisations aux ressources définies de la manière suivante :

> *"At any point in time, the total utilization or consumption of a specified resource by all vehicles must be less than or equal to a specified limit."*

Cette phrase définit avec précision la deuxième problématique de notre étude. Pour cela, nous expliquons la notion de ressources de notre problème.

L'entreprise est constituée d'un ensemble de sites industriels différents :

- carrière : pour la fourniture en matières premières (graviers, sable) ;

- plateforme : pour le stockage des matières premières et des matériaux à recycler ;
- centrale d'enrobés : pour la production d'enrobés à chaud et à froid ;
- chantier : pour la réalisation d'un ouvrage de travaux publics.

Cette liste n'est pas exhaustive mais elle représente les sites majeurs du réseau de transport de l'entreprise.

Sur chacun des sites cités ci-dessus, des ponts-bascules pèsent les camions pour assurer un suivi logistique des matériaux transportés. Ces ponts-bascules peuvent peser un seul camion à la fois. Ainsi, le nombre de camions pouvant utiliser ce pont-bascule est limité par une capacité d'accueil. Cette capacité est vue comme une contrainte.

Le pont-bascule est typiquement la représentation d'une ressource sur laquelle les véhicules doivent être synchronisés. Mais également, nous pouvons considérer comme ressource toutes les machines servant au chargement (chargeuse, pelleteuse) ou déchargement (finisseur) des camions.

Les problèmes de tournées de véhicules où il existe des synchronisations aux ressources sont plus difficiles à résoudre puisque les contraintes de synchronisation viennent complexifier le problème. C'est un champ relativement nouveau de notre discipline introduit en 2008 par Irnich [60].

La synchronisation aux ressources constitue le deuxième verrou scientifique de cette thèse.

1.4 Plan de la thèse

Afin de répondre aux deux problématiques énoncées dans la section précédente, nous avons établi le plan de recherche ci-après.

Le chapitre 2 présente le problème industriel que nous résolvons. Il introduit un ensemble de notions et de notations clés pour la compréhension de cette thèse. Une formulation mathématique du problème global est également proposée.

Le chapitre 3 présente une revue de la littérature des problèmes de tournées de véhicules que nous avons consultée au cours de notre recherche. Les problèmes de collectes et de livraisons y sont détaillés. Nous proposons une étude bibliographique des problèmes connexes à notre recherche en détaillant notamment les problèmes de tournées en camions complets et les problèmes à flotte hétérogène de véhicules. La synchronisation dans les problèmes de tournées de véhicules est également présentée.

Le chapitre 4 présente la méthodologie générale de résolution du problème de cette thèse. Nous y présentons les motivations ainsi que les approches métiers issues de l'entreprise et prises en compte dans ces travaux. Le fonctionnement de notre algorithme y est décrit.

Le chapitre 5 présente la première phase de résolution de notre problème. Il s'agit d'une phase de découpage qui affecte des demandes aux camions. Les enjeux de cette affectation sont détaillés dans une première partie. Nous présentons ensuite deux modèles de résolution : (i) monopériodique et (ii) multipériodique.

Le chapitre 6 présente la méthode que nous utilisons pour découper les demandes en requêtes (correspondant à un camion complet) de transport. Ce découpage permet de prendre en compte les différents types de demandes de notre problème : (i) les demandes à flux tendus et (ii) les demandes à flux détendus. Celui-ci est spécifique à notre problème pour modéliser fidèlement les contraintes de l'entreprise.

La recherche de solutions à notre problème est présentée dans le chapitre 7. Nous détaillons l'algorithme que nous utilisons pour trouver des solutions au problème. Une méthode heuristique est proposée. Nous introduisons une nouvelle méthode de vérification des contraintes temporelles pour vérifier la réalisabilité d'une solution. La méthode est évaluée sur des instances de la littérature et sur des instances de l'entreprise.

Le chapitre 8 de recherche est consacré à la résolution d'un problème étendu. Dans la première partie de ce chapitre, nous présentons des méthodes pour la minimisation de la durée des tournées. Ces méthodes sont innovantes et testées sur les mêmes instances. Enfin, dans la deuxième partie, nous présentons une méthodologie heuristique pour prendre en compte les pauses-déjeuner des chauffeurs dans les tournées.

Le dernier chapitre 9 récapitule les travaux présentés dans cette thèse. Des pistes de recherche et perspectives de travail futures sont discutées.

I

Le problème de collectes et livraisons avec fenêtre de temps et synchronisation sur les ressources : description et positionnement dans la littérature

Table des matières

Dans cette partie, nous aborderons deux chapitres. Le chapitre 2 présente une description du problème dans sa globalité en introduisant les notions élémentaires et les notations utilisées dans ce manuscrit. Le chapitre 3 propose une revue de la littérature sur les problèmes de tournées de véhicules qui sont proches du nôtre et qui nous ont guidés pour la compréhension et la résolution de notre problème.

2

Description du problème de tournées en camions complets

Sommaire

Dans ce chapitre, nous présentons le problème de planification des tournées de véhicules, correspondant au problème opérationnel de l'entreprise. Le problème de transport, dans sa globalité, est présenté dans la section 2.1. La section 2.2 présente les enjeux scientifiques. Enfin, une formulation mathématique du problème, base de notre recherche, est proposée dans la section 2.3.

2.1 Description du problème

Le problème de tournées de camions complets présenté dans cette thèse est un problème de transport réel issu d'une entreprise du secteur des travaux publiques. Les entreprises de ce secteur doivent organiser

quotidiennement la planification des véhicules pour réaliser l'ensemble des tâches à effectuer. Dans notre cas, il s'agit de planifier les tournées d'une flotte hétérogène de véhicules de manière à satisfaire un ensemble d'ordres de transport. Certaines contraintes industrielles, logistiques, sociétales et légales viennent complexifier ce problème. Dans cette section, nous présentons les différentes caractéristiques du problème considéré.

2.1.1 Périodes

Le problème à résoudre est défini sur un horizon temporel d'au plus une semaine. Cet horizon temporel est divisé en plusieurs périodes. Lorsque l'horizon de temps comporte une et une seule période, nous appelons cet horizon de temps monopériodique. Dans le cas contraire, où l'horizon de temps comporte plusieurs périodes, l'horizon de temps est dit multipériodique.

La version monopériodique du problème correspond typiquement à une journée de travail de l'entreprise. L'horizon de temps et la période sont alors confondus. Dans le cas d'un problème multipériodique, l'horizon de temps, typiquement une semaine, comporte plusieurs périodes. Chacune de ces périodes est assimilée à un jour de la semaine. Dans le cas de travaux de nuit, des périodes supplémentaires peuvent être créées. Dans la version multipériodique, deux périodes ne peuvent pas se chevaucher.

2.1.2 Sites

Les activités de l'entreprise se déroulent sur plusieurs sites géographiques où des véhicules font transiter des produits. Les sites ont différentes fonctions sur les produits notamment : (i) de production, (ii) de stockage, (iii) d'utilisation et (iv) de recyclage. De plus, certains sites, que nous appelons dépôts, sont destinés spécifiquement au stockage des véhicules. Le tableau 2.1 présente les différentes opérations effectuées sur les différents sites de notre problème.

Dénomination du site	Produits				Véhicules
	Produire	Stocker	Utiliser	Recycler	Stocker
Chantier	•	•	•		
Carrière	•				
Centrale d'enrobés	•	•	•	•	
Plateforme		•	•	•	
Centrale de recyclage		•		•	
Site d'enfouissement		•			
Fournisseur	•				
U.I.O.M.[1]		•	•		
Dépôt					•

Tableau 2.1 – Description des opérations pouvant être effectuées sur les différents sites de l'entreprise.

Pour chaque période et chaque site, l'horaire est délimité par une heure d'ouverture et une heure de fermeture. En dehors de cette plage horaire, l'accès au site est impossible. La capacité de stockage en produits des sites est considérée infinie. Chaque site peut accueillir un certain type de véhicules en fonction du gabarit de ceux-ci ou bien d'autres contraintes d'accès. De la même façon, en fonction de la nature du site, celui-ci ne peut disposer que d'un sous ensemble de produits ce qui a une influence sur la définition des ordres de transport.

Deux notions importantes sont à définir : (1) capacité de flux et (2) durée de service.

[1] Usine d'Incinération d'Ordures Ménagères

Capacité de flux Chaque site a une capacité horaire limitée et finie sur le flux de véhicules pouvant circuler sur un site. Sur la plupart des sites, un pont-bascule est présent pour enregistrer le passage en entrée et en sortie d'un véhicule. Le plus souvent, c'est l'élément déterminant car il n'autorise le passage que d'un seul véhicule à la fois. Cette capacité impose une contrainte sur les transports qui est décrite plus précisément dans la section 2.2. La capacité de flux est exprimée en nombre maximum de véhicules par unité de temps. Lorsqu'il n'y a pas de limite sur le flux de véhicules pouvant circuler dans un site, cette capacité sera infinie.

Durée de service Chaque véhicule effectue une opération sur un site, généralement un chargement ou un déchargement de produits. Le contrôle de cette opération se fait sur le pont-bascule. La durée de contrôle dépend de la nature du pont-bascule et de son fonctionnement (manuel ou automatisé). Cette durée passée sur le pont-bascule et en opération est appelée durée de service.

2.1.3 Produits et flux

Chaque activité de l'entreprise de travaux publics nécessite des produits. Ces produits, de nature différente, servent à élaborer les différentes phases d'un chantier. Deux groupes de produits sont à considérer : (1) les produits vracs et (2) les produits finis.

Les produits vracs sont un groupe de produits où le dénombrement est impossible. L'expression du besoin pour ces produits se fait uniquement sous forme de masse exprimée en tonnes (t). Nous distinguons plusieurs sous-groupes parmi ces produits dont les principaux sont les enrochements, les graviers, les sables, les bétons et les enrobés. Les trois premiers sous-groupes possèdent des granulométries, des densités différentes en fonction de leurs utilisations. Les deux derniers sont des produits préparés par l'entreprise.

Les produits finis peuvent être dénombrés et comptabilisés en tonnes ou en unités. Dans ce groupe, nous retrouvons, par exemple, les buses, les bordures ou bien encore les gaines. Exceptionnellement, ces produits peuvent être transportés sur des produits vracs mais ce type de mélange ne rentre pas dans le cadre de notre étude.

Nous distinguons également la manière dont les produits sont utilisés. Les flux tendus sont définis lorsque la collecte et/ou la livraison d'un produit doit se faire de manière continue. Les collectes ou livraisons de deux camions pour un même ordre de transport doivent être successives et sans délais. Par opposition, les flux sont dits détendus lorsque les livraisons peuvent être effectuées à n'importe quel moment de la période sans contrainte de succession.

Cas des enrobés Les enrobés font partie des flux tendus. Ils servent à élaborer la couche supérieure du revêtement des chaussées. Pour pouvoir appliquer les enrobés, le produit doit être chaud. Il est donc collecté à la centrale d'enrobés sous forme chaude et livré sans délai pour une application manuelle (à la pelle) ou bien mécanisée (au finisseur). Dans le cas d'une application mécanisée, le finisseur, qui applique les enrobés, doit être approvisionné constamment en produits pour ne pas créer de malfaçons dans la construction routière (crevasses ou renflements). Une bonne régularité de la surface de la route est due à une bonne régularité dans l'application des enrobés. C'est pour cela que ce produit est considéré comme un flux tendu. Plus spécifiquement, il existe un temps maximum d'utilisation entre la sortie de la centrale d'enrobé et son application, généralement de trois heures.

Un autre exemple de flux tendu est l'évacuation d'un produit. Nous pouvons citer l'exemple d'une pelle mécanique qui évacue une quantité de terre de manière constante (débit horaire constant) en la mettant dans des camions. Les camions doivent alors évacuer (collecter) le produit de manière continue pour ne pas créer de discontinuité dans l'ordre de transport d'évacuation.

2.1.4 Véhicules

Les entreprises de travaux publics utilisent différents types de véhicules pour la réalisation des travaux. Les véhicules servent à transporter des produits, à effectuer des opérations spécifiques (finisseur, tracto-pelle), à transporter d'autres véhicules (porte-véhicules) ou encore à transporter des travailleurs. Dans notre étude, nous ne retiendrons que les véhicules servant à transporter des produits. Deux catégories principales composent cette flotte de véhicules : les semi-remorques et les camions-bennes. Les véhicules des autres catégories ne sont utilisés que trop marginalement pour présenter un potentiel d'optimisation.

Les semi-remorques sont des véhicules composés d'un camion (partie motorisée) qui tracte une remorque (partie non motorisée) par l'intermédiaire d'une sellette. Les remorques sont indépendantes du camion et possèdent leurs propres essieux. Elles sont interchangeables entre les camions. Dans notre étude, pour chaque période donnée, le semi-remorque conserve la remorque qui lui est affectée. Deux types de semi-remorques sont présents dans la flotte de véhicules.

Les camions-bennes sont similaires aux semi-remorques mais la différence majeure est la benne qui est portée par le camion sur un plateau. La benne ne possède donc pas d'essieux. Il existe quatre types de camions-bennes différents.

Globalement, les semi-remorques (SR) offrent des capacités de transport (charges utiles) en produits supérieures à celles des camions-bennes (CB). Ces capacités, exprimées en tonnes, sont reportées dans le tableau 2.2.

Désignation	SR	CB	capacité	nombre d'essieux
SEMI43	•		30	-
SEMI38	•		25	-
8X4		•	16	4
6X4		•	12	3
4X2		•	10	2

Tableau 2.2 – Présentation de la flotte de véhicules : nom de la catégorie, capacité exprimée en tonne, nombre d'essieux.

Indifféremment, nous utiliserons le terme camion ou véhicule dans la suite de cette étude. Les contenants (remorques ou bennes) peuvent être différents avec des caractéristiques spécifiques : matière de fabrication (aluminium ou acier), couverte (bâchée ou non), mode de déchargement (bi-benne ou tri-benne). Toutes ces caractéristiques propres à chacun des contenants créent des compatibilités avec les produits pouvant être transportés. Le transport des enrobés se fera exclusivement dans des bennes ou remorques bâchées pour conserver le plus longtemps la température chaude pour l'application. Les enrochements sont transportés dans des bennes en acier, plus solides que les bennes en aluminium. Le mélange de deux produits dans une benne ou une remorque est interdit. Chaque contenant reçoit un et un seul produit pour chaque transport.

Chaque catégorie possède un nombre connu de véhicules au moment de la résolution du problème. La catégorie d'appartenance d'un camion peut limiter le passage sur certains axes du réseau routier (lien avec le nombre d'essieux). Les véhicules ne circulent pas à la même vitesse entre chaque catégorie.

De plus, l'utilisation de chaque véhicule engendre un coût financier. Nous considérons que les coûts identiques pour tous les véhicules d'une catégorie, qui sont : (i) coût fixe d'utilisation sur une période, (ii) coût d'utilisation horaire et (iii) coût d'utilisation kilométrique.

Enfin, lorsque la flotte de camions est insuffisante pour résoudre un problème, l'entreprise peut utiliser une flotte de camions en sous-traitance avec des conditions tarifaires différentes (tarif journalier ou tarif horaire).

Au vue de ces caractéristiques différentes, la flotte de véhicule est dite hétérogène et de capacité finie.

2.1.5 Chauffeurs

Les véhicules de l'entreprise sont conduits par des chauffeurs salariés de l'entreprise. À travers la planification des transports, le problème considéré doit prendre en compte la législation liée au transport routier et au secteur des travaux publics. Les règles de conduite des chauffeurs sont définies par :

- la législation européenne ;
- la législation française ;
- le code du travail ;
- la convention collective.

Chaque chauffeur est affecté à un véhicule en début de période et cette affectation reste unique pour toute la période considérée. Parmi les règles de conduite existantes, nous en retiendrons deux :

- le temps de conduite journalier ne doit pas dépasser 9h ;
- une interruption de 45 min est obligatoire toutes les 4h30 de conduite continue.

De plus, dans le cas de travail sur une journée, notre étude devra proposer une coupure pour le déjeuner du midi d'une durée de 1h30 définie dans une fenêtre de temps et ayant lieu sur un des sites de l'entreprise.

2.1.6 Demandes de transport

Pour satisfaire les prestations contractuelles de l'entreprise, celle-ci découpe ses travaux (ou chantiers) en ordres de transport.

Ces ordres de transport (ou demandes dans la suite de cette thèse) sont l'expression d'une quantité donnée d'un produit à transporter entre un point d'origine (ou point de collecte) vers un point de destination (ou point de livraison). Ce transport devra s'effectuer dans une fenêtre de temps précisée. Sur le point de collecte (respectivement le point de livraison), le camion doit charger (resp. décharger) le produit.

Plusieurs sites, proposant le même produit, peuvent être l'origine de la collecte d'une demande. Dans notre cas d'étude, le choix des origines et destinations des ordres de transport est effectué avant la phase d'optimisation et est considéré comme une donnée d'entrée.

Les demandes en produits, exprimées en tonnes, nécessitant d'être transportées sont généralement nettement supérieures à la capacité des camions de la flotte. Ainsi, pour la réalisation d'une demande, plusieurs camions peuvent être nécessaires. Chaque transport élémentaire d'un camion satisfaisant une partie ou la totalité d'une demande est appelé requête. Cette requête s'effectue en camions pleins. Ainsi, une demande est constituée de la somme d'une à plusieurs requêtes.

2.1.7 Notion de ressource

Nous avons défini dans les parties précédentes, les notions de demandes, requêtes et sites industriels. Certaines opérations des camions s'effectuent sur des sites industriels tels que des chantiers. L'existence d'un pont-bascule, dans certains cas, limite le flux permis dans un espace de temps. Dans d'autre cas, l'utilisation d'un outil externe (e.g. un finisseur pour l'application des enrobés) limite également le flux de camions pouvant effectuer un chargement ou déchargement. D'autres exemples sont l'utilisation d'une pelle mécanique pour le chargement, une trémie de chargement, un quai de déchargement.

Cette notion de limite ou d'accès restreint est modélisée par la notion de ressource. Une ressource limite le nombre de véhicules pouvant effectuer une opération à un même instant sur un même site. Cette notion a été formalisée par Drexl [35] comme *resource synchronization* (vu dans la section 1.3).

Dans notre problème, nous considérons des ressources de capacité unitaire. Un seul camion à la fois peut exécuter une opération, sur une ressource, seul ou en utilisant un outil externe. Dans le cadre d'une

demande en flux tendu, une ressource spécifique sera affectée à cette demande. Et sur cette demande, toutes les requêtes s'effectueront sans discontinuité temporelle, c'est à dire sans *time lags* entre le service de deux requêtes successives.

Dans notre étude, un camion occupera la ressource durant une certaine durée définie comme la durée de service sur la ressource. Cette durée d'occupation de la ressource est différente de la durée totale nécessaire au camion pour effectuer une opération. La durée d'occupation de la ressource est équivalente à la durée d'utilisation de la ressource par le camion. Nous supposons que l'ordre des camions ne change pas sur le site de la ressource.

Par exemple, pour une livraison de matériau sur une plateforme de stockage, un camion va se peser en entrée et en sortie de la plateforme sur le pont-bascule. La durée de service sur la ressource pont-bascule sera de deux fois le temps de pesée. Entre ces deux pesées, le camion déchargera son contenu. La durée du déchargement est définie comme la durée du service au sein de sa tournée.

2.1.8 Description d'une solution du problème

Nous définissons une solution du problème étudié comme un ensemble de tournées pour chaque camion de la flotte. Cette solution doit respecter l'ensemble des contraintes du problème. Une tournée est définie par la succession des points visités par un camion lors d'une période. Durant ce trajet, le camion va collecter et livrer un certain nombre de requêtes en respectant les contraintes du problème. Une tournée peut éventuellement être vide si le camion ne travaille pas dans cette période. Une pause-déjeuner devra être planifiée dans un intervalle donné si le camion travaille sur une amplitude journalière. Chaque requête devra fournir les fenêtres de temps de service des demandes associées.

Pour évaluer la qualité d'une tournée, nous utiliserons un indicateur économique composé des termes suivants :

- le coût des transports ;
- le coût de la masse salariale.

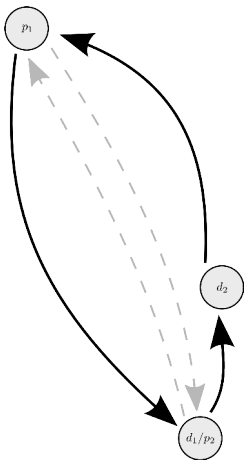
2.2 Positionnement scientifique de notre recherche

La particularité des transports en travaux publics est le volume de produits à transporter entre les différents sites d'opérations. Pour la réalisation d'ouvrages routiers comme d'ouvrages de construction, de grandes quantités de produits, sous forme de vrac dans la grande majorité des cas, sont nécessaires et doivent être acheminées d'un site de production à un site d'utilisation.

Les pratiques courantes de la logistique actuelle dans les travaux publics font qu'un sous-ensemble de camions est affecté à chaque demande. Pour acheminer le volume total de produits d'une demande, ces camions vont effectuer un certain nombre d'aller-retours, appelés rotations, entre le site de chargement et le site de déchargement (chantier). Les allers sont effectués à plein, camion chargé au maximum de sa capacité par le produit et les retours à vide. Ces trajets à vide représentent un coût non négligeable pour une entreprise de travaux publics où, dans notre cas, le nombre de kilomètres parcourus à l'année se compte en millions.

Les techniques d'optimisation et la connaissance du réseau de l'ensemble des demandes peuvent permettre des mutualisations de trajets pour diminuer une partie des trajets à vide. Un des premiers enjeux de cette étude est de diminuer le nombre de kilomètres à vide parcourus par les camions. Pour cela, l'objectif est de profiter du trajet retour à vide des camions pour transporter les produits d'une autre demande qui se situerait dans la même zone géographique.

L'application des enrobés étant sujet à des contraintes particulières, il est nécessaire de prévoir plusieurs rotations de camions pour satisfaire une demande. L'utilisation d'une ressource, typiquement un finisseur pour l'application des enrobés, est fortement contraint temporellement par les requêtes qui devront y être

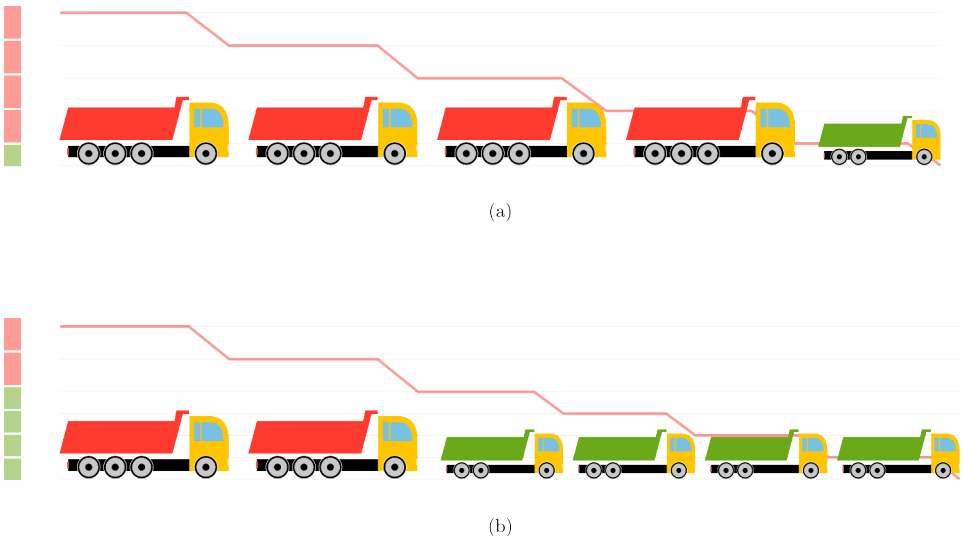


Exemple 1 – Deux demandes 1 et 2 sont représentées sur cette figure. Le point de livraison d_1 de la demande 1 est identique au point de collecte p_2 de la demande 2. Mutualiser (livrer la demande 2 immédiatement après la demande 1) permet de réduire deux trajets à vide représentés en pointillé.

affectées. Le flux continu d'approvisionnement en enrobés génère un flux continu de requêtes à positionner sur la ressource et de surcroît affectés à des camions différents.

La quantité de produits nécessitant d'être transportée pour chaque demande excédant la capacité des camions, il est nécessaire d'avoir plusieurs camions pour satisfaire la livraison totale d'une demande. La flotte de véhicules est hétérogène, il existe donc potentiellement plusieurs combinaisons de camions de différentes capacités pour effectuer cette demande. De plus, la durée de déchargement dans ce cas dépend fortement de la capacité du camion.

La flotte de camions de l'entreprise est connue et les camions ne sont disponibles qu'un certain nombre d'heures par période (disponibilité fonction du temps de travail des chauffeurs). Ainsi, l'affectation des camions aux demandes n'est pas triviale mais est le résultat d'un processus décisionnel. Nous parlerons de découpage des demandes en requêtes pour ce deuxième enjeu. Une explication illustrée du découpage est proposée dans la figure 2.



(a)

(b)

Exemple 2 – Cette figure montre deux possibilités de découpage d'une demande de 70 tonnes. Deux types de camions sont à disposition : petit camion de capacité de 10 tonnes et grand camion de capacité de 15 tonnes. La solution (a) utilise 4 grands camions et 1 petit camion tandis que la solution (b) utilise 2 grands camions et 4 petits camions.

Dans les problèmes standards de tournées de véhicules, chaque livraison peut être traitée individuel-

lement. Dans notre cas, ces livraisons doivent être coordonnées. Cette caractéristique est nommée par le terme synchronisation. Nous pouvons distinguer deux types de synchronisation.

La première synchronisation concerne la livraison des demandes à flux tendus. Pour ce type de demandes, plusieurs livraisons sont nécessaires et doivent être effectuées sans interruption. Une requête a effectuée par un camion est dite synchronisée avec une autre requête b quand il existe une relation temporelle de précédence ou succession entre ces deux livraisons. L'heure h_b de livraison de la requête b devra être égale à l'heure h_a de la requête a plus la durée de service s_a nécessaire de la requête a.

$$h_b = h_a + s_a$$

La deuxième synchronisation a un lien avec les ressources. Concrètement, cette synchronisation s'établit sur un point de passage obligatoire aux sites de l'entreprise : le pont-bascule. Sur un pont-bascule, les requêtes seront synchronisées pour satisfaire une relation entre chacune d'entre elles et la ressource concernée. La tournée d'un camion peut-être assimilée à une potentielle consommation de ressources à chaque opération. Pour notre cas d'étude, les ressources sont de capacité unaire, un seul camion peut être servi à la fois sur chaque ressource.

Ces deux types de synchronisation sont présentés dans la figure 3.

(a) Exemple de synchronisation à une ressource. Un seul véhicule peut accéder au pont-bascule à la fois.

(b) Exemple de synchronisation entre deux camions. Le camion vert doit attendre la fin du déchargement du camion rouge pour effectuer son opération de déchargement.

Exemple 3 – Deux types de synchronisation.

2.3 Formulation du problème

2.3.1 Notations

Dans cette partie, nous présentons la formulation mathématique du problème. Pour cela, nous introduisons un ensemble de notations mathématiques pour décrire le problème. Les contraintes précises du problème sont spécifiées au fur et à mesure de la description.

Nous considérons un graphe $G_0 = (V_0, A_0)$ associé au problème. V_0 est l'ensemble des nœuds du graphe et A_0 l'ensemble des arcs. L'ensemble des points de collecte des demandes est noté P^D et l'ensemble des points de livraison est noté D^D. L'ensemble des dépôts des véhicules est noté O avec O_1 l'ensemble des dépôts de départ et O_2 l'ensemble des dépôts d'arrivée. Chaque dépôt $o \in O$ possède une fenêtre de temps $[e_o, l_o]$ durant laquelle un véhicule peut y accéder. L'ensemble des nœuds de pauses-déjeuner est noté L.

Notre problème considère un ensemble de catégories de véhicules noté F. Pour chaque catégorie de véhicules $f \in F$, il existe n^f véhicules de capacité Q^f. L'ensemble des différents véhicules est noté K. Un véhicule $k \in K$, de capacité q^k, commence sa tournée au dépôt d'origine $o_1(k) \in O_1$ et finit sa tournée au dépôt d'arrivée $o_2(k) \in O_2$. Lorsqu'un véhicule $k \in K$ de la flotte est utilisé durant une période, on lui associe un coût fixe $\mathcal{CD}^k$ d'utilisation, un coût kilométrique $\mathcal{CK}^k$ et un coût horaire $\mathcal{CH}^k$.

Nous considérons un ensemble N de demandes de transport. Pour chaque demande $n \in N$, il existe un site géographique de collecte et un site géographique de livraison. La quantité de produit de la demande n est notée q_n.

Comme la quantité d'une demande excède généralement la capacité nominale des véhicules, il est nécessaire d'effectuer plusieurs requêtes pour satisfaire la livraison totale d'une demande. L'ensemble des requêtes de toutes les demandes est nommé R. L'ensemble des points de collecte des requêtes est noté P_0 et l'ensemble des points de livraison est noté D_0. Pour chaque demande n, l'ensemble des requêtes associé à cette demande est R^n. On notera pour une requête $r \in \{1, \ldots, |R|\}$, le point de collecte $p_r \in P_0$ et le point de livraison $d_r \in D_0$. L'ensemble des requêtes, pour une demande n, est présenté dans la figure 2.1.

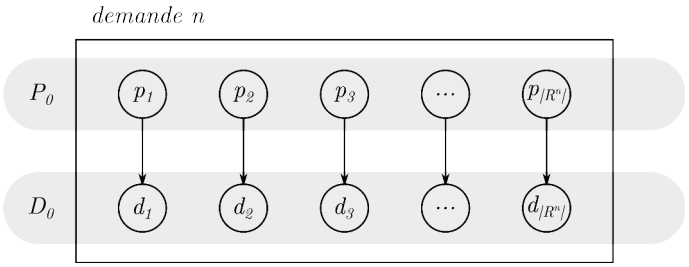


FIGURE 2.1 – Modélisation de l'ensemble des sommets des points de collecte et de livraison associées aux requêtes de la demande n.

La durée de service par un véhicule $k \in K$ pour l'opération au point de collecte p_r (respectivement au point de livraison d_r) est noté $s^k_{p_r}$ (resp. $s^k_{d_r}$). Pour chaque point $i \in P_0 \cup D_0$, nous définissons une fenêtre de temps $[e_i, l_i]$ dans laquelle l'opération peut commencer.

Pour simplifier la compréhension de la modélisation, nous introduisons également l'ensemble $K^n \subset K$ représentant l'ensemble des véhicules compatibles avec la demande $n \in N$. L'ensemble des demandes à flux tendus est noté $N_{\mathcal{S}} \subset N$.

Le réseau de notre graphe comprend des arcs entre certaines paires de nœuds. Le distancier entre chaque nœud est dépendant de la nature du véhicule étant donné leurs différences de vitesse de transport. Il en est de même pour le coût d'utilisation d'un arc entre deux nœuds. Pour chaque paire de nœuds $(i, j) \in V^2$ et chaque véhicule k, nous notons par t^k_{ij} le temps de trajet du nœud $i \in V$ vers le nœud $j \in V$ effectué par le véhicule $k \in K$. La distance de ce même arc par le même véhicule est notée d^k_{ij}.

Une pause-déjeuner est effectuée entre un point de livraison et un point de collecte. Le début d'une pause-déjeuner $l \in L$ doit avoir lieu dans la fenêtre de temps $[e_{lunch}, l_{lunch}]$.

L'ensemble des nœuds du graphe est $V_0 = P_0 \cup D_0 \cup O_1 \cup O_2 \cup L$. Le problème est défini sur l'ensemble des arcs suivants : $A_0 = \{(o_1(k), o_2(k)) | k \in K\} \cup O_1 \times P_0 \cup \{(p_r, d_r) | r \in R\} \cup D_0 \times O_2 \cup D_0 \times P_0 \cup D_0 \times L \cup L \times P_0$.

2.3.2 Variables

Plusieurs variables sont associées à notre problème et représentent les décisions à prendre. Les variables de décision x^k_{ij} sont des variables binaires qui valent 1 si le véhicule $k \in K$ emprunte l'arc $(i, j) \in A_0$, 0 sinon. Pour exprimer les contraintes du problème, on définit les variables intermédiaires suivantes : l'heure de début de service d'un nœud $i \in V_0$ est représentée par la variable h_i. La variable binaire y_k vaut 1 si le véhicule k est utilisé dans la solution, 0 sinon.

2.3.3 Fonction objectif

L'objectif de notre problème dans un premier temps est de minimiser le coût total de transport comprenant le coût fixe d'utilisation des véhicules, le coût d'utilisation des arcs du réseau par les véhicules et le coût du temps de travail.

Nous appelons $\mathscr{F}$ la fonction objectif associée au problème, avec x l'ensemble des variables du problème :

$$\mathscr{F}(\mathrm{x}) = \sum_{k\in K} \mathcal{CD}^k * y^k + \sum_{k\in K}\sum_{(i,j)\in A} \mathcal{CK}^k * x_{ij}^k * d_{ij}^k + \sum_{k\in K} \mathcal{CH}^k * (h_{o_2(k)} - h_{o_1(k)}) \tag{2.1}$$

Le problème peut être modélisé comme un problème de collectes et livraisons avec fenêtres de temps Pickup and Delivery Problem with Time Windows (PDPTW). Nous choisissons de présenter les contraintes du problème par famille. Certaines notations sont introduites au fur et à mesure de la modélisation.

2.3.4 Contraintes

Les contraintes traditionnelles d'un problème PDPTW sont les suivantes [38] :

$$\sum_{(o_1(k),i)\in A_0} x_{o_1(k)i}^k = \sum_{(i,o_2(k))\in A_0} x_{io_2(k)}^k \qquad \forall k \in K \tag{2.2}$$

$$\sum_{(o_1(k),i)\in A_0} x_{o_1(k)i}^k = y^k \qquad \forall k \in K, i \neq o_2(k) \tag{2.3}$$

$$\sum_{(i,j)\in A_0} x_{ij}^k = \sum_{(j,i)\in A_0} x_{ji}^k \qquad \forall k \in K, \forall j \in V_0 \tag{2.4}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A_0, \forall k \in K \tag{2.5}$$

$$y_k \in \{0,1\} \qquad \forall k \in K \tag{2.6}$$

Les contraintes (2.2) et (2.3) obligent chaque véhicule à commencer sa tournée à son dépôt de départ et la finir à son dépôt d'arrivée et lient les variables x_{ij}^k et y^k. Un véhicule k inutilisé dans la solution emprunte l'arc $(o_1(k), o_2(k))$. La conservation des flux est assurée par les contraintes (2.4).

Les contraintes relatives aux fenêtres de temps sont définies de la manière suivante :

$$e_i \leq h_i \leq l_i \qquad \forall i \in V_0 \tag{2.7}$$

$$h_j \geq (h_i + s_i + t_{ij}^k) x_{ij}^k \qquad \forall (i,j) \in A_0, \forall k \in K \tag{2.8}$$

$$h_i \in \mathbb{R}^+ \qquad \forall i \in V \tag{2.9}$$

Les contraintes (2.7) imposent que l'heure de début de service soit inclue dans la fenêtre de temps. Les contraintes (2.8) assurent la cohérence des fenêtres de temps pour deux points successifs lors de la tournée d'un véhicule et peuvent être linéarisées en utilisant les techniques standards [24].

La modélisation de la satisfaction d'une demande $n \in N$ est basée sur la création d'un ensemble R^n de requêtes de transports associées à cette demande. Le nombre maximal de requêtes nécessaires à cette satisfaction est déterminé par le véhicule de plus petite capacité et compatible avec la demande. Cette capacité est notée $Q_{\min}^n = \min_{k\in K} q^k$. Ainsi, le nombre maximal de requêtes associées à une demande n est borné par $\frac{q_n}{Q_{\min}^n}$.

Dans une solution du problème, certaines requêtes de l'ensemble R^n peuvent ne pas être utilisées. Une variable binaire u_r est introduite au modèle et vaut 1 si la requête r est utilisée dans la solution. Nous définissons également une fonction ς qui, pour chaque couple $(n,i)|n \in N, i \in 1,\ldots,|R^n|$, $\varsigma(n,i)$ retourne la requête qui est à la position i dans l'ensemble R^n. Nous définissons un ordre sur l'ensemble R^n *a priori*.

$$\sum_{r\in R^n}\sum_{k\in K^n} x_{p_r d_r}^k q^k \geq q_n \qquad \forall n \in N \tag{2.10}$$

$$u_r = \sum_{k\in K} x_{p_r d_r}^k \qquad \forall r \in R \tag{2.11}$$

$$u_r \in \{0,1\} \qquad \forall r \in R \tag{2.12}$$

Les contraintes (2.10) imposent la satisfaction de chaque demande. Les contraintes (2.11) lient les variables u_r et les variables x_{ij}^k.

L'ensemble des ressources est appelé Π. L'ensemble des nœuds qui effectuent leurs opérations sur une ressource π est noté $V_\pi \subset P \cup D$. Une seule opération peut être effectuée à la fois sur chaque ressource. Les variables binaires z_{ij}^π sont introduites et valent 1 si le nœud $i \in V_\pi$ est servi avant le nœud $j \in V_\pi$ sur la ressource π.

$$h_j \geq (h_i + s_i) z_{ij}^\pi \qquad \forall \pi \in \Pi, \forall i \in V_\pi, \forall j \in V_\pi \tag{2.13}$$
$$z_{ij}^\pi + z_{ji}^\pi = 1 \qquad \forall \pi \in \Pi, \forall i \in V_\pi, \forall j \in V_\pi \tag{2.14}$$
$$z_{ij}^\pi \in \{0, 1\} \qquad \forall \pi \in \Pi, \forall i \in V_\pi, \forall j \in V_\pi \tag{2.15}$$

Les contraintes (2.13) et (2.14) permettent de modéliser les contraintes entre requêtes utilisant une même ressource. Les contraintes (2.14) signifient qu'il existe un seul ordre entre deux nœuds sur une ressource.

Cette modélisation permet de prendre en compte toutes les demandes du problème. En revanche, pour les demandes à flux tendus, il convient de définir plus précisément l'occupation des ressources. Dans le cadre d'une demande à flux tendu $n \in N_S$, il peut y avoir un flux tendu au point de collecte, au point de livraison ou bien aux deux. L'ensemble $N_S^p \subset N_S$ (respectivement $N_S^d \subset N_S$) représente les demandes ayant une contrainte de flux tendu sur le point de collecte (resp. le point de livraison).

$$u_{\varsigma(n,1)} = 1 \Rightarrow h_{p_{\varsigma(n,1)}} = e_{p_{\varsigma(n,1)}} \qquad \forall n \in N_S^p \tag{2.16}$$
$$u_{\varsigma(n,i)} = 1 \Rightarrow h_{p_{\varsigma(n,i)}} = h_{p_{\varsigma(n,i-1)}} + s_{p_{\varsigma(n,i-1)}} \qquad \forall n \in N_S^p, i > 1 \tag{2.17}$$
$$u_{\varsigma(n,1)} = 1 \Rightarrow h_{d_{\varsigma(n,1)}} = e_{d_{\varsigma(n,1)}} \qquad \forall n \in N_S^d \tag{2.18}$$
$$u_{\varsigma(n,i)} = 1 \Rightarrow h_{d_{\varsigma(n,i)}} = h_{d_{\varsigma(n,i-1)}} + s_{d_{\varsigma(n,i-1)}} \qquad \forall n \in N_S^d, i > 1 \tag{2.19}$$
$$u_{\varsigma(n,1)} = 1 \qquad \forall n \in N_S \tag{2.20}$$
$$u_{\varsigma(n,i)} \geq u_{\varsigma(n,i+1)} \qquad \forall n \in N_S \tag{2.21}$$

Les contraintes (2.16) imposent que pour chaque demande la première requête commencent au début de la fenêtre de temps de la collecte. Les contraintes (2.17) renforcent l'exécution des requêtes au point de collecte pour les demandes à flux tendus. Le service d'une requête i doit être effectué après le service de la requête $i-1$ plus sa durée de service associé. Idem pour les contraintes (2.18) et (2.19) au point de livraison. Les contraintes (2.20) indiquent qu'il existe au moins une requête qui doit être servie la première. Deux requêtes consécutives sont liées par les contraintes (2.21). Les contraintes avec des implications peuvent être linéarisées en utilisant les techniques standards du grand M.

Remarque pour le lecteur Dans la réalité, ces requêtes auront des fenêtres de temps réduites dans le cadre des requêtes à flux tendus. Ceci est du à la capacité du camion qui servira la requête et qui définit une fenêtre de temps proportionnelle à sa capacité. Pour des soucis de simplicité du modèle, ici, nous supposons que les fenêtres de temps sont égales à celles de la demande.

Le temps de conduite journalier ne doit pas dépasser une durée T^{daily} de 9h. Le fonctionnement standard d'une journée de travail dans l'entreprise considérée comprend une séparation de la période en deux demi-journées par une pause-déjeuner. Cette pause-déjeuner, d'une durée s_{lunch} doit être planifiée pour tous les chauffeurs à l'intérieur de la fenêtre de temps suivante $[e_{lunch}, l_{lunch}]$. Chaque véhicule k effectue une pause-déjeuner au nœud $l^k \in L^k$ avec $L^k \subset L$.

$$h_{o_2(k)} - h_{o_1(k)} \leq T^{daily} - s_{lunch} \qquad \forall k \in K \tag{2.22}$$

$$\sum_{i \in D} x_{ij}^k = y^k \qquad \forall k \in K, \forall j \in L \tag{2.23}$$

$$\sum_{i \in D} x_{ij}^k = 1 \Rightarrow h_j \in [e_{lunch}, l_{lunch}] \qquad \forall k \in K, \forall j \in L \tag{2.24}$$

Les contraintes (2.22) imposent le respect d'une durée maximale pour les tournées. Pour chaque véhicule, les contraintes (2.23) imposent le passage du véhicule et de son chauffeur par une pause-déjeuner. Les contraintes (2.24) imposent que la pause-déjeuner ait lieu dans sa fenêtre de temps.

Nous avons présenté une modélisation de notre problème global en nous attardant sur la modélisation liée aux ressources, aux demandes et aux requêtes. Dans les prochains chapitres, nous reviendrons en détail sur cette modélisation en fonction des contraintes abordées dans ces chapitres.

3

Revue de la littérature

Sommaire

Le problème résolu dans le cadre de cette thèse fait partie de la catégorie des problèmes de tournées de véhicules Vehicle Routing Problem (VRP) [102]. Cette famille de problèmes a reçu beaucoup d'attention dans la littérature depuis plusieurs décennies. De nombreuses études académiques et industrielles ont découlé du problème de voyageur de commerce Traveling Salesman Problem (TSP) [29] dont font partie les problèmes de tournées de véhicules. Nous nous intéressons aux problèmes de VRP, lorsque les demandes à servir se situent sur des nœuds, par opposition aux problèmes de tournées de véhicules sur arcs Arc Routing Problem (ARP) où les requêtes se situent sur des arcs ou segments de réseaux.

Pour guider le lecteur dans cette revue de la littérature, nous proposons dans un premier temps, dans la section 3.1, d'étudier le problème de collectes et livraisons Pickup and Delivery Problem (PDP) en particulier sa généralisation au cas avec fenêtres de temps Pickup and Delivery Problem with Time Windows (PDPTW). Nous nous attarderons principalement sur les articles évoquant le transport de matériaux relatif au secteur des travaux publics et du transport de bois. Dans un second temps, nous proposons une synthèse

de la littérature portant sur des problèmes connexes et proches de notre étude dans la section 3.2. Enfin, nous présenterons, dans la section 3.3, la synchronisation et les aspects temporels dans les problèmes de collectes et livraisons. La section 3.4 conclue cette revue de la littérature.

3.1 Le problème de collectes et livraisons

Le problème de collectes et livraisons avec fenêtres de temps PDPTW est une généralisation du problème de collectes et livraisons PDP ainsi que du problème de tournées de véhicules avec fenêtres de temps Vehicle Routing Problem with Time Windows (VRPTW). Dans cette section, nous présentons le PDPTW ainsi que des applications particulières pour les problèmes de transports de matériaux dans le secteur des travaux publics et du bois.

Un des premiers articles présentant le PDPTW a été écrit par Dumas et al. [38]. Le problème consiste à satisfaire un ensemble de requêtes de transport caractérisées par une quantité de produit à livrer, d'un point de collecte à un point de livraison, avec des contraintes de fenêtre de temps et de précédence entre les points de collecte et les points de livraison. Une méthode exacte, utilisant la génération de colonnes pour le problème maître et une méthode de plus court chemin avec contraintes pour le sous-problème, permet de résoudre des instances de 19 à 55 requêtes. Aujourd'hui, des instances de plusieurs centaines de requêtes sont résolues à l'optimum [9]. Trois revues de la littérature [32], [12], [34] font état des dernières tendances et méthodes de résolution pour résoudre des problèmes dérivés du PDPTW. En particulier, Mitrovic-Minic [75] présente une revue de la littérature comportant différentes méthodes exactes et approchées de résolution pour les problèmes de PDP avec les fenêtres de temps. Une notation unifiée pour classer ces problèmes est proposée par Berbeglia et al. [14], comportant trois champs : [Structure|Visites|Véhicules]. Suivant cette classification, notre problème est classé par le triptyque suivant [1-1|P-D|m]. Le premier champ *structure* 1-1 signifie que chaque requête a une seule origine et une seule destination. Le deuxième champ signifie que l'opération de collecte et celle de livraison sont distinctes et sur deux points différents tandis que le dernier champ indique l'utilisation de m véhicules. Une classification différente est proposée par Parragh et al. [80] en différenciant les problèmes avec des interactions ou non entre clients et dépôts. Dans cette classification, notre problème correspond au problème de tournées de véhicules avec collecte et livraison Vehicle Routing Problem with Pickup and Delivery (VRPPD) où l'origine et la destination des requêtes sont connues a priori.

3.1.1 Méthodes de résolution exactes

Des méthodes exactes pour résoudre le PDPTW ont été proposées. Une méthode de type *Branch and Cut* est présentée par Ropke et al. [86]. Deux modélisations avec des variables à deux indices sont formulées pour résoudre le problème. Cinq familles d'inégalités valides sont décrites, dont trois nouvelles et viennent enrichir les coupes apportées à la résolution. L'ensemble de ces composants permet à l'algorithme de résoudre à l'optimalité des instances de huit véhicules et jusqu'à 96 requêtes.

Cette méthode a été étendue à un algorithme de type *Branch and Cut and Price* par Ropke et Cordeau [85]. Le sous-problème correspond à un problème de plus court chemin avec des contraintes de fenêtre de temps, de capacité et de collectes et livraisons (Elementary Shortest Path Problem with Time Windows, Capacity, and Pickup and Delivery : ESPPTWCPD). Les mêmes instances que Ropke et al. [86] sont résolues sous une limite de temps de dix minutes. Dans un temps alloué de deux heures, trois instances sur six, comportant 500 requêtes, sont résolues à l'optimalité.

Baldacci et al. [9] proposeront deux années plus tard un algorithme à base de *Branch and Price* résolvant plus d'instances à l'optimalité. En effet, 15 instances sont améliorées avec des temps de calcul neuf fois plus rapides en moyenne.

3.1.2 Méthodes approchées de résolution

Parallèlement aux méthodes exactes, des méthodes approchées ont été proposées pour résoudre des instances comportant un nombre plus important de requêtes. Parmi les premiers travaux publiés, nous pouvons citer celui de Nanry et Barnes [76]. Ils ont proposé une méthode de type tabou réactif (Reactif Tabu Search : RTS) où les paramètres de l'algorithme peuvent être modifiés à chaque itération. Trois opérateurs de voisinage sont utilisés pour explorer l'espace des solutions : (i) déplacer une requête collecte / livraison d'une tournée vers une autre tournée, (ii) échanger deux requêtes de deux tournées différentes et (iii) déplacer une requête au sein de sa tournée. Leur méthode est testée sur un jeu d'instances modifiées de Solomon comportant entre 25 et 100 requêtes. Le temps moyen de résolution des instances varie de la seconde pour les instances de 25 requêtes jusqu'à trois minutes pour les instances de 100 requêtes.

Li et Lim [65] proposent une méthode de type recuit simulé Simulated Annealing (SA) avec une liste tabou qui mémorise les solutions déjà obtenues. L'objectif est de minimiser la somme de plusieurs termes pondérés : le nombre de véhicules, la distance parcourue, la durée de la solution et les temps d'attente. Les auteurs autorisent leur algorithme à redémarrer si un nombre donné d'itérations a été effectué sans produire aucune amélioration de la solution courante. Pour tester leur méthode, ils ont créé un nouveau jeu de données de référence de 56 instances, comprenant 100 requêtes, en se basant sur les instances de Solomon.

Dans les méthodes de résolution utilisant la notion de voisinage, les algorithmes effectuent des modifications de la solution (par ajout ou retrait de requêtes) afin de trouver une meilleure solution. Le coût d'insertion de requêtes dans une solution, modifiant ainsi la valeur de la fonction objectif, est un critère commun d'évaluation d'une insertion. Lu et Dessouky [70] proposent d'associer à ce coût, l'augmentation du temps de trajet et la réduction des marges disponibles aux fenêtres de temps. Cet ajout permet de contrer l'effet aveugle d'une insertion uniquement basée sur le coût d'insertion. De plus, les auteurs utilisent une heuristique de meilleure insertion basée sur le nombre de croisements créés au sein de chaque tournée (Crossing Length Percentage : CLP). Pour des temps d'exécution plus rapides que ceux de Li et Lim [65] (13 secondes en moyenne contre 730 secondes en moyenne), l'algorithme produit des solutions de bonne qualité toutefois sans égaler leurs résultats (en moyenne 8% pires).

Une décomposition du problème en deux parties est proposée par Bent et Van Hentenryck [13]. La première partie a pour objectif de réduire le nombre de véhicules dans la solution. Pour cela, une méthode à base de recuit simulé SA s'exécute durant cinq minutes maximum. Un opérateur de voisinage déplace une paire de nœuds choisie aléatoirement. La fonction objectif de cette première partie minimise trois objectifs lexicographiquement : (i) le nombre de tournées, (ii) le nombre de tournées avec peu de clients et (iii) le coût de trajet. La deuxième partie de l'algorithme minimise le coût des tournées en utilisant une méthode de recherche à voisinage large Large Neighborhood Search (LNS) inspirée de la méthode proposée par [99]. Un unique opérateur de voisinage déplace au plus p paires de clients. Le paramètre p varie au cours de la recherche ainsi que le nombre d'itérations. L'algorithme est testé sur un grand nombre d'instances de PDPTW et trouve un grand nombre de nouvelles solutions.

Des améliorations significatives des résultats sur les instances de Li et Lim sont obtenues par Ropke et Pisinger [87]. La méthode utilisée est une recherche adaptative à voisinage large Adaptive Large Neighborhood Search (ALNS). Cette méthode alterne successivement entre le retrait et l'ajout de requêtes dans une solution en utilisant plusieurs opérateurs de destruction et de reconstruction. Plusieurs opérateurs sont proposés pour chacune des étapes. À chaque itération de l'algorithme, le choix d'utiliser un opérateur est basé sur sa performance lors des précédentes itérations. L'acceptation d'une solution est pilotée par un recuit simulé SA. Presque toutes les instances de Li et Lim sont améliorées en terme de nombre de véhicules utilisés et de distance parcourue dans la solution finale. 48 nouvelles instances sont créées avec un nombre de requêtes allant jusqu'à 500. La méthode ALNS présente de meilleures solutions sur les instances de test que la méthode LNS avec des temps de calcul inférieurs. Les temps de résolution augmentent néanmoins significativement avec le nombre de requêtes de l'instance.

D'après la revue de la littérature, nous remarquons que les problèmes de PDPTW sont plus difficiles à

résoudre que les problèmes de VRP, notamment à cause des contraintes de précédences entre collecte et livraison [12]. Les méthodes heuristiques proposées permettent de résoudre des problèmes de grandes tailles dans des temps de calcul raisonnables, mais de nombreuses perspectives de recherche restent d'actualité sur le développement de méthodes exactes et heuristiques pour résoudre ce type de problèmes.

3.1.3 Transport d'enrobé

Le problème décrit dans la section 2.1 présente, entre autre, le transport des enrobés (Asphalt Concrete : AC) dans le secteur des travaux publics. La littérature sur le transport de ce produit est limitée mais quelques travaux ont néanmoins été recensés.

Le travail de master de Brachner [17] étudie la construction et la maintenance d'ouvrages routiers. Le problème décrit l'approvisionnement en enrobés d'un ou plusieurs chantiers. Une seule livraison d'enrobés est nécessaire par chantier et ceux-ci sont appliqués par des équipes de travailleurs. La flotte de véhicule est considérée homogène et la durée de service associée au déchargement des enrobés est nul. Un modèle de programmation linéaire en variables mixtes Mixed Integer Programming (MIP) est résolu grâce au solveur GUROBI pour des instances de petites tailles. Pour des instances de taille plus réaliste, une heuristique de type *Multi-Start-Greedy* est proposée.

Pour une entreprise norvégienne de travaux publics, Rubasheuskaya [88] propose plusieurs modélisations mathématiques d'un problème combiné de transport et ordonnancement. Les chantiers considérés ont une demande qui peut excéder la capacité des camions. Pour connaître le nombre de camions et le type de camions à affecter à un chantier, un découpage du chantier en livraisons par type de camions est effectué au préalable.

Les deux articles mentionnés ci-dessus sont les seuls traitant uniquement de la livraison d'enrobés à notre connaissance.

3.1.4 Transport de béton

Les problèmes de tournées de véhicules avec application d'enrobé sont assez proches des problèmes de livraisons de béton (*concrete* en anglais). Les ordres de transport associés à la livraison de ces deux produits sont généralement supérieurs à la capacité des camions. La livraison des bétons par camions toupies doit se faire également dans une fenêtre de temps restreinte pour éviter les risques de prise du béton. Pour les enrobés, il est nécessaire d'avoir un flux continu afin de ne pas créer de rupture dans l'approvisionnement, typiquement dans le cas où un finisseur est utilisé. Dans le cadre des bétons, il est nécessaire de respecter un temps court entre deux livraisons successives même si la livraison en flux continu n'est pas une nécessité.

Dans ses travaux de thèse, Durbin [39] s'intéresse à la livraison de béton pour une entreprise aux États-Unis. La motivation de son étude est le temps d'attente recensé par l'entreprise au point de chargement et de déchargement des toupies de béton, de capacités égales dans l'étude. Le gain annuel lié à la diminution de cinq minutes au point d'attente est estimé entre 500 000$ et 750 000$. Un outil d'aide à la décision est proposé et traite le problème de façon dynamique en proposant des méthodes pour ajuster les quantités de béton à livrer et les tournées de transport en temps réel. L'auteur utilise une représentation espace-temps pour résoudre le problème en utilisant une modélisation de type *minimum-cost network flow* résolue par une méthode tabou pour déterminer les horaires des chauffeurs.

Les travaux de thèse de Schmid [94] étendent les travaux de Durbin [39] pour la modélisation du problème sous forme de Minimum Cost Flow Problem (MCNF). Le problème industriel traité dans la thèse de Schmid [94] concerne la livraison de béton où les demandes sont connues a priori et en utilisant une flotte hétérogène de véhicules. Le temps entre deux livraisons consécutives pour une même demande est minimisé dans la fonction objectif (et non posé comme une contrainte). Deux approches de résolution sont testées :

- une première approche hybride combinant un solveur de type MIP et une méthode de type Variable Neighborhood Search (VNS). La première approche est présentée dans Schmid et al. [95]. Elle com-

bine l'utilisation d'une méthode VNS et d'un solveur MIP intégrés à une méthode de type Variable Large Neighborhood Search (VLNS) pour résoudre le problème. Deux utilisations hybrides sont proposées : collaborative et intégrée. Le VNS permet d'améliorer durant un certain temps la solution en utilisant six opérateurs de *shaking* et trois opérateurs d'améliorations. Dans le VLNS, la phase de *shaking* libère les valeurs de certaines variables de décisions, l'exécution d'une requête pour une demande ou bien l'utilisation d'un camion pour une requête, dont la valeur est ensuite laissée pour décision au solveur MIP. Sur des instances de taille moyenne, des solutions (avec un gap de 12% par rapport à la borne inférieure) sont trouvées dans des temps de 2500 secondes en moyenne.

- une deuxième méthode combinant un modèle MCNF utilisant des patrons associés à un VNS (un patron est défini comme la succession de véhicules pour livrer une demande de béton). La deuxième approche est décrite dans une autre publication de Schmid et al. [96]. Pour cette résolution, le réseau est discrétisé en périodes de cinq minutes suivant la formulation proposée par Durbin et Hoffman [40]. Des patrons sont créés pour résoudre le problème. L'algorithme détermine pour chaque véhicule un temps de passage. Le modèle MCNF est résolu en utilisant un MIP sur un ensemble restreint de patrons pour conserver des temps de résolutions courts. Ensuite, une méthode VNS utilisant un opérateur de *shaking* (changement de véhicules entres différents patrons) et une recherche locale est utilisée pour améliorer et changer la structure de la solution. Cette méthode vient créer de nouveaux patrons qui sont résolus une nouvelle fois par le solveur. Jusqu'à dix boucles sont ainsi effectuées. Les résultats obtenus par cette deuxième approche se révèlent moins prometteurs que la première approche.

Asbach et al. [8] considèrent la livraison de béton pour une entreprise de distribution de béton en utilisant une flotte hétérogène de véhicules. Une modélisation en variables mixtes est proposée pour déterminer toutes les livraisons de béton par tous les types de véhicules. Le problème résultant est de trop grande dimension pour être résolu par un solveur, c'est pourquoi les auteurs proposent une méthode à base de recherche locale. Cette méthode relaxe certaines variables du problème dans un premier temps, puis résout le problème entier avec une heuristique d'insertion au meilleur coût dans un second temps. Ce processus est répété durant un temps alloué d'une heure. Les résultats montrent que les instances de tests révèlent une file d'attente aux dépôts d'approvisionnement durant la matinée de la journée de travail. Cependant, certaines instances ne sont pas résolues, des demandes n'étant pas satisfaites, dans le temps de calcul alloué.

La même modélisation mathématique est proposée par Liu et al. [69]. Leur problème traite la construction de tournées pour des camions de béton ainsi que le placement de pompes à béton au début de chaque site de construction. Un algorithme génétique est proposé pour résoudre le problème et est comparé à une méthode, basée sur la simulation et combinant différentes règles de priorités, précédemment utilisée par les mêmes auteurs [68]. La nouvelle méthode propose une meilleure solution sur une instance réelle d'une journée comprenant 8 camions, 2 pompes et l'approvisionnement de 7 chantiers.

Cette littérature des problèmes de transport de bétons montre bien la difficulté du problème à résoudre. En effet, la planification doit souvent être reconsidéré, pour une actualisation de l'ordre de la minute, en fonction de l'état des livraisons en bétons. Au regard de la taille des problèmes à résoudre, les solveurs actuels sont souvent limités par le temps de calcul alloué pour fournir une solution de très bonne qualité. Les méthodes proposées ci-dessus sont performantes pour des instances de tests de taille limitée mais ne sont guère applicables pour des instances industrielles. Le développement de métaheuristiques est une bonne approche pour les futurs travaux.

3.1.5 Transport de bois

Le transport de bois par camions Log Truck Scheduling Problem (LTSP) est un problème industriel qui comporte un grand nombre de similitudes avec notre problème. En effet, les transports de bois depuis le lieu d'extraction (forêt) vers le lieu de transformation (usine) nécessitent le déplacement de grandes quantités de bois grâce à l'utilisation du transport terrestre. Pour approvisionner les usines de transformation de bois,

beaucoup d'allers-retours sont nécessaires et une machine est utilisée pour charger et décharger les grumes de bois. Les pays scandinaves (Suède, Norvège, Finlande), le Chili, la Nouvelle-Zélande et le Canada ont été à l'initiative de beaucoup d'études sur ce sujet. Le problème LTSP consiste à décider des flux d'approvisionnement entre les différents lieux d'extraction et les différents lieux de transformation.

Pour pallier une insuffisance d'organisation du réseau de transport de bois au Chili, Weintraub et al. [104] proposent un modèle de simulation avec des heuristiques de choix pour décider du transport de bois d'une des huit plus grandes exploitations forestières du Chili. Leur algorithme propose une résolution journalière du problème d'affectation des camions aux demandes de transport. Cet outil a permis de réduire de 20% les coûts logistiques d'exploitation avec une discrétisation du temps de 15 minutes. Trois minutes sont nécessaires pour proposer une solution à un problème utilisant 220 camions, 40 origines et 15 destinations.

Pour résoudre un problème d'affectation entre des forêts et des usines, Palmgren et al. [77] proposent une méthode basée sur la génération de colonnes pour un problème d'une entreprise suédoise comportant 28 camions, 266 demandes et 15 destinations. La génération de schémas types de tournées réalisables se fait en utilisant une méthode de type *Cluster First Route Second*. La méthode ne permet pas de trouver de solution optimale après quatre heures d'exécution mais propose néanmoins une solution plus courte en terme de distance parcourue de 8% par rapport à une solution de l'entreprise faite à la main où les demandes ont d'abord été séparées par région géographique. Ce travail a été étendu par les mêmes auteurs dans [78] toujours en utilisant la génération de colonnes. Le sous-problème est résolu avec un algorithme de k plus courts chemins (*k-shortest paths algorithm*). En une heure de calculs avec l'utilisation du solveur CPLEX, les auteurs résolvent un problème comprenant 187 demandes, 15 points de livraisons et utilisant une flotte de véhicules homogènes de 28 camions. Le problème n'est pas résolu jusqu'à l'optimalité mais, après un certain nombre d'itérations, la solution fournie est à 8% de la borne inférieure du problème.

Une version simplifiée du LTSP est le Timber Transport Vehicle Routing Problem (TTVRP) où les lieux d'origines et de destinations des requêtes de transport sont connus a priori. Ce problème est décrit par Hirsch [57]. Les transports sont faits exclusivement en camions pleins avec des contraintes de compatibilités entre les camions et les tournées pouvant être empruntées. Une formulation en variables mixtes est proposée et résolue par le solveur Xpress-MP pour des instances de petites tailles. Pour de plus grandes instances, une méthode de recherche tabou incluant trois variantes est proposée. Les variantes sont des combinaisons de la méthode tabou standard [23] et de voisinages plus ou moins larges basés sur le retrait d'arcs correspondant à des transports à vide. La méthode offre des solutions de très bonne qualité (déviation à la meilleure solution obtenue inférieure à 1 %) pour 100 000 itérations de la méthode dans un temps de dix minutes. Les résultats sont confirmés sur deux jeux de 20 instances comprenant 30 ordres de transport et dix camions. Les travaux sont par la suite étendus par Gronalt et Hirsch [55].

Pour résoudre un problème LTSP sur un horizon de temps hebdomadaire, Flisbert et al. [45] proposent une méthode en deux phases. La première phase résout un problème d'affectation des producteurs aux transformateurs. Les demandes non servies durant la période concernée sont pénalisées dans la fonction objectif. Cette première phase permet de créer tous les flux de transport et revient à résoudre un problème de type TTVRP dans la deuxième phase. Cette deuxième phase est résolue par une méthode tabou étendue (Extended Unified Tabu Search Algorithm : EUTSA). Des contraintes de fenêtres de temps, de dépôts multiples, de demandes sur plusieurs périodes enrichissent le problème. Cette méthode permet de résoudre une instance réelle de 110 véhicules qui n'est pas résolue manuellement. Sur des instances plus petites (dix véhicules), la méthode propose jusqu'à 30% d'amélioration par rapport à la solution manuelle du logisticien. Néanmoins, le problème ne considère pas l'attente des camions sur les sites de production et de transformation.

Une version du LTSP avec synchronisation des camions et d'appareils pour le chargement et le déchargement est présentée par El Hachemi et al. [44]. Une méthode en deux phases comportant la résolution d'un problème d'affectation pour la première phase et un problème de tournées pour la deuxième phase est proposée. La première phase est résolue par le solveur SCIP. Cette phase crée les flux de transport entre les forêts et les transformateurs de bois en respectant les capacités des différents sites. Elle résout un problème

hebdomadaire d'affectation qui est transformé en un problème LTSP journalier. Pour résoudre la deuxième phase, une méthode hybride combine un solveur de contraintes (COMET) avec une méthode de recherche locale itérative à base d'opérateurs de voisinage combinée à un algorithme d'ordonnancement au plus tôt. L'objectif est de minimiser les temps d'attente et la date de fin du planning. Les tests sont effectués sur des instances de 400 et 700 demandes de transport par semaine. La combinaison d'une méthode à base de contraintes et de recherche locale itérative augmente la qualité de la solution jusqu'à 14% par rapport à une méthode de recherche locale simple. Ces travaux sont étendus dans El Hachemi et al. [43] pour résoudre le même problème avec les contraintes de temps de pause des chauffeurs. Les auteurs proposent une discrétisation du temps par pas de 20 min, ce qui correspond au temps moyen des opérations de chargement ou de déchargement. La deuxième phase de ce problème est résolue par le solveur CPLEX. Une stratégie de branchement sur les variables de temps permet de réduire le temps d'attente total de la solution. Les résultats sur les instances de El Hachemi et al. sont améliorées en moyenne de 4 %.

Les méthodes de résolution utilisées pour les problèmes de transport de bois constituent une approche intéressante pour notre problème. Comme pour notre problème, des contraintes de synchronisation peuvent exister, cependant la flotte de véhicules est généralement homogène ce qui constitue une différence essentielle. Les travaux les plus récents proposent des résultats de bonne qualité mais ne permettent pas encore de gérer finement les transports avec des pas de temps plus précis. Une revue de la littérature est présentée par Rönnqvist et al. [90] sur les récents problèmes de logistiques pour le transport de bois. Les auteurs présentent les derniers travaux et proposent 33 challenges ouverts pour les industriels et les chercheurs. Parmi ceux-là, nous pouvons citer le développement de méthode incluant la synchronisation entre les camions et les chargeurs ou le développement de méthodes de résolution incluant des modifications de planning en temps réel.

3.2 Problèmes connexes

Dans cette partie, nous présentons d'autres problèmes de la littérature qui partagent certaines caractéristiques avec notre problème. Nous retiendrons quatre catégories de caractéristiques : (1) le transport en camions pleins, (2) la caractérisation de la flotte de véhicules, (3) le découpage des demandes et (4) le transport de conteneurs.

3.2.1 Problème de transport en camions pleins

Le transport en camions pleins est une des caractéristiques de notre problème de tournées de véhicules. La littérature est moins riche sur cette thématique bien que les problèmes soient courants dans les transports industriels. Par camion plein, on définit le transport d'une marchandise d'un point de collecte vers un point de livraison, où le camion n'est autorisé à transporter que cette marchandise sur le trajet. Ces problèmes peuvent être modélisés par des arcs de transport a_{ij} indiquant la quantité de produits ou la quantité de camions nécessaires à transporter du point i vers le point j.

Un des premiers cas d'étude concerne une entreprise de l'industrie chimique désirant transporter ses produits avec sa flotte de camions complétée de camions en sous-traitance. Ce problème a été traité par Ball et al. [11] et par Bodin et al. [15] en 1983. Trois heuristiques sont proposées : deux de type *Route-first, Cluster-second* et une méthode gloutonne. Ces travaux sont les prémices de la littérature ayant trait aux transports en camions pleins.

Quelques années plus tard, un algorithme exact utilisant une méthode de *Branch and Bound* a été proposé par Desrosiers et al. [33]. Les arcs où sont nécessaires plusieurs transports entre deux points sont remplacés par autant de sommets. Les auteurs cherchent à minimiser le coût des transports à vide. L'algorithme proposé résout à l'optimalité des petites instances. La qualité de la solution est fortement impactée par le nombre de dépôts et le nombre de véhicules présents par dépôt.

Arunapuram et al. [7] introduisent le problème de tournées de véhicules avec camions pleins Vehicle Routing Problem with Full Truckloads (VRPFL). Le problème présenté est très proche du Multiple-Depot

Vehicle Scheduling Problem (MDVSP), la différence étant la quantité de camions pleins nécessaire pour effectuer une demande, quantité unitaire dans le MDVSP et supérieure à un dans le VRPFL. Il s'agit de minimiser les trajets à vide des camions d'une flotte homogène de véhicules répartis sur plusieurs dépôts. Les auteurs proposent un algorithme de *Branch and Price*. Pour cette modélisation basée sur les chemins, les auteurs montrent que le temps d'exécution de l'algorithme ne dépend pas du nombre de dépôts qui varie de un à dix. En revanche, la largeur des fenêtres de temps a une influence importante sur les temps d'exécution puisqu'il existe plus de chemins possibles dans le sous-problème du *Branch and Price*.

Parallèlement aux travaux de Arunapuram et al. [7], Gronalt et al. [54] proposent une formulation linéaire basée sur les chemins du Full Truckloads Pickup and Delivery Problem with Time Windows (FTPDPTW) . La fonction objectif minimise également les transports à vide. Chaque point doit être visité dans sa fenêtre de temps et les tournées doivent respecter une durée maximale. Quatre versions d'un algorithme de résolution basé sur la méthode des écarts de Clarke and Wright [22] sont étudiées. Les auteurs montrent également l'effet de la largeur des fenêtres de temps sur la qualité de la solution en soulignant que des fenêtres de temps plus larges amènent des solutions avec moins de transport à vide.

Pour une entreprise britannique de travaux publics, Currie et Salhi [27] modélisent le problème de livraison de camions pleins pour l'élaboration de tournées comme un Full Load PDPTW. Le problème considère des fenêtres de temps, des compatibilités entres véhicules et produits ainsi qu'une flotte hétérogène de véhicules. Seulement les instances de six véhicules et 50 requêtes sont résolues à l'optimalité avec un solveur en 24h de temps de calcul. Une méthode hybride associant un algorithme glouton et de regret avec trois opérateurs de voisinages est proposée. Ces trois opérateurs seront, par la suite, utilisés dans une méthode tabou (Currie et Salhi [28]) pour trouver de meilleures solutions aux instances réelles fournies par l'entreprise. Lorsqu'une demande est supérieure à la capacité des camions, un découpage préliminaire est effectué et donné par l'entreprise avant la phase d'optimisation. Ces travaux sont relativement similaires au nôtre mais ne considèrent pas la gestion de la synchronisation entre demandes.

Liu et al. [67] résolvent un problème Multi-Depot Capacitated Arc Routing Problem with Full Truckloads (MDCARPFL). Une flotte de véhicules homogènes de capacité Q doit livrer un ensemble de requêtes de transport dont les quantités sont supérieures à Q. Le découpage des ordres de transport en requêtes se fait grâce à un Programmation Linéaire en Nombre Entiers (PLNE). Ensuite, une méthode heuristique en deux phases est proposée. La première phase vient construire un ensemble de cycles, rattachés ou non, à un dépôt. La deuxième phase combine ces cycles en utilisant la meilleure insertion comme critère. Même si les fenêtres de temps ne sont pas prises en compte, l'algorithme résout des instances jusqu'à 300 demandes.

Récemment, Venkateshan et Mathur [103] ont étendu le problème VRPFL pour une flotte de véhicules hétérogènes. Dès lors, la minimisation du coût des trajets à vide ne suffit plus puisqu'il faut aussi s'intéresser au coût des transports, qui dépendent de la nature des véhicules. Pour résoudre le sous problème du problème maître, de nouvelles règles de parcours de graphe sont proposées en tenant compte du fait qu'un véhicule peut visiter plusieurs fois un client. La méthode est prometteuse mais encore lente (deux heures de calcul) pour des instances avec 20 demandes.

Les problèmes de transport en camions pleins sont des problèmes relativement difficiles à résoudre puisqu'ils traitent souvent de demandes qui excèdent la capacité des camions. La taille des instances pouvant être résolues à l'optimalité dans des temps relativement courts ([7]) témoigne de cette difficulté. Ainsi, ce sont surtout des méthodes heuristiques qui résolvent les problèmes de taille réelle. Aucun des problèmes cités ne comporte toutes les caractéristiques de notre problème (voir tableau 3.1).

3.2.2 Flotte hétérogène de véhicules

Une des principales variantes du VRP est le problème avec flotte hétérogène de véhicules, où chaque véhicule peut avoir une capacité ou un coût d'utilisation différent. Le réseau de transport associé au problème peut aussi avoir plusieurs temps de trajets variant en fonction du véhicule. Les véhicules identiques peuvent être regroupés par famille. Le problème a été présenté en premier lieu par Golden et al. [51] où la dénomination Fleet Size and Mix problem (FSM) traite des problèmes où il faut décider du nombre de véhicules de

chaque famille à utiliser (supposant la flotte illimitée). Lorsque la flotte de véhicules est donnée, le problème est nommé : problème de tournées de véhicules avec flotte hétérogène Heterogeneous Vehicle Routing Problem (HVRP). Ce problème a été introduit par Baldacci et al. [10], ainsi qu'une notation en trois parties pour catégoriser ces problèmes : (i) taille de la flotte limitée (HVRP) ou illimitée (FSM), (ii) coût fixe d'un véhicule (F) ou non (∅) et (iii) coût des trajets dépendants des véhicules (D) ou non (∅). Pour notre problème, nous retiendrons la dénomination Heterogeneous VRP with Fixed Costs and Vehicle-Dependent Routing Costs (HVRPFD). Cette classe de problèmes traite d'une flotte limitée de véhicules avec des coûts associés à chaque véhicule et des temps de trajets dépendant de la nature du véhicule.

Pour résoudre le HVRP, Gendreau et al. [47] ont utilisé la méthode tabou avec l'heuristique GENIUS (efficace pour le TSP) ainsi que des opérateurs de voisinages permettant notamment de choisir une flotte de véhicules de coût plus élevé pour diversifier la recherche. Une autre méthode heuristique à base de voisinage a également été utilisée par Prins [84] pour résoudre le HVRP : l'heuristique, inspirée de Clarke and Wright [22], fusionne deux tournées de camions en respectant notamment des contraintes de temps de trajet. Sur une instance réelle, l'heuristique a permis d'obtenir un gain de 11% correspondant à une économie annuelle de 2 000 000$.

Le HVRPFD a été résolu avec de bonnes performances par Li et al. [66]. Les auteurs ont proposé une méthode *Multistart Adpative Memory Programming* (MAMP) basée sur une méthode tabou, des opérateurs de voisinages (2-opt, relocate, swap) ainsi qu'un *path relinking*. Les solutions aux instances de Taillard [101] sont presque toutes améliorées avec des temps de calcul divisés par dix.

Une méthode exacte de type *Branch and Cut and Price* est enfin proposée par Pessoa et al [81] pour résoudre le HVRP. L'algorithme fonctionne rapidement (environ cinq minutes) pour 9 instances jusqu'à 100 clients résolues à l'optimum.

Les problèmes cités ci-dessus incluent des contraintes supplémentaires, par rapport au VRP, relatives à la flotte de véhicules. Le problème reste toutefois difficile à résoudre et les applications et les méthodes proposées ne traitent pas du cas où la demande est largement supérieure à la capacité des véhicules. De plus, le choix de l'affectation des types de camions aux demandes n'est pas pris en compte.

3.2.3 *Split Delivery*

Le *split delivery* (SD) est la possibilité de fractionner (*split* en anglais) une demande pour la servir par plus d'un véhicule. Cette notion a été introduite par Dror et Trudeau [36]. La solution fournie alors en autorisant le SD est meilleure en terme de distance parcourue. Une méthode tabou est proposée par Archetti et al. [4] et améliore de 5% les résultats des instances de 15 clients de Dror et Trudeau [36]. Les tests sont effectués sur des instances jusqu'à 100 clients avec des temps de calcul de l'ordre de la dizaine de minutes.

Le SD présente l'avantage de minimiser le nombre de véhicules nécessaires en séparant les demandes sur plusieurs véhicules. Une revue de la littérature a été récemment publiée sur le Split Delivery Vehicle Routing Problem (SDVRP) par Archetti et Speranza [3].

3.2.4 Transport de conteneurs

Un problème proche du transport de marchandises en camions complets est le transport de conteneurs. En effet, les camions transportent des conteneurs depuis un lieu de collecte vers un lieu de livraison. Caris et Janssens [20] modélisent le transport de conteneurs à l'intérieur un terminal d'expédition comme un FTPDPTW. Pour résoudre leur problème, les auteurs proposent une méthode en deux phases. Dans la première phase, des paires de clients (collectes et livraisons) sont constituées en respectant les contraintes de fenêtres de temps et un temps maximum de service entre les deux clients. Ces paires sont triées suivant quatre critères pondérés (temps de trajet, marge disponible, opportunité de la paire au regard du temps de trajet, opportunité de la marge disponible). La deuxième phase élabore des tournées en se servant des véhicules triés par ordre décroissant de coût. Cette solution initiale est ensuite améliorée par trois opérateurs de voisinage (CROSS, COMBINE, INSERT). Des instances générées aléatoirement de 100 et 200 clients

sont résolues en dix minutes environ avec des écarts par rapport à la borne inférieure ne dépassant pas 0.5%. Pour transporter des conteneurs homogènes entre des terminaux, Imai et al. [59] proposent une heuristique à base de relaxation lagrangienne. La contrainte représentant la durée maximale d'une tournée est relaxée dans la fonction objectif. Le problème ainsi résolu est équivalent à un problème généralisé d'affectation (General Assignment Problem : GAP). Les résultats ont été testés sur 432 instances aléatoires et les auteurs obtiennent des résultats à moins de 10% de la borne inférieure dans des temps de calcul inférieurs à deux minutes.

3.3 Synchronisation et contraintes temporelles

Nous présentons dans cette section les aspects temporels qui peuvent exister dans les problèmes de tournées de véhicules. Dans un premier temps, nous introduisons la problématique de synchronisation. Dans un second temps, nous présenterons la prise en compte des contraintes temporelles sur les problèmes de tournées de véhicules.

3.3.1 Problème avec synchronisation

Les problèmes de tournées de véhicules avec synchronisation sont devenus une classe de problème émergente au cours des dix dernières années. La revue de la littérature de Drexl [35] propose une classification des problèmes relevant de cette thématique. La synchronisation est identifiée quand plus d'un véhicule est nécessaire pour la réalisation d'une tâche de transport. Quatre classes de problèmes sont distinguées de la manière suivante :

- *les synchronisations de tâches* définissent les tâches qui doivent être servies une et une seule fois par un ou plusieurs véhicules.
- *les synchronisations d'opérations* sont définies quand le temps écoulé Δ entre les débuts de service des opérations respectives de deux véhicules doit être compris dans un intervalle $[a, b]$ où $a \leq b$. Quand la présence d'un véhicule A est nécessaire pour mouvoir un autre véhicule B, nous parlons de *synchronisation de mouvement*.
- *la synchronisation de chargement* est quand la somme des flux de chargement et déchargement d'un produit à un point donné est égale à la demande de ce point.
- *les synchronisations aux ressources* sont définies lorsqu'il existe une limite pour l'utilisation simultanée d'une ressource à un point donné par les véhicules.

Pour cette dernière catégorie de synchronisations, le principe a été introduit par Hempsch et Irnich [56] sous le nom de *inter-tour resource constraint*. Les auteurs considèrent trois types de ressources : (i) chargement, (ii) temps et (iii) coût. La solution du problème est représentée sous la forme d'un grand tour. Des opérateurs de voisinage (2-opt, swap, relocation) standards sont utilisés dans une méthode de type LNS. Pour s'échapper d'un minimum local, plusieurs clients sont enlevés aléatoirement. Pour plus d'informations sur les problèmes de synchronisation, nous invitons le lecteur à se référer à l'état de l'art de Drexl [35].

Pour résoudre un problème de transport de personnes Dial A Ride Problem (DARP), Masson et al. [72] modélisent le problème sous forme d'un Dial-A-Ride Problem with Transfers (DARPT). Un point de transfert est alors défini pour permettre le transfert de personnes d'un véhicule à un autre. Ce problème est résolu avec une méthode de type ALNS en développant des opérateurs spécifiques pour gérer la notion de transfert. Les opérateurs *transfer point removal*, *pickup/delivery cluster removal* et *history removal* sont présentés et adaptés avec la notion de transfert. De même, les opérateurs de meilleure insertion *best insertion with transfer*, *transfer first* et de regret *regret insertion with transfer* sont proposés pour la reconstruction de solutions. Les auteurs montrent que l'ajout de synchronisations a un effet positif sur la qualité des solutions et peut amener un gain jusqu'à 8% sur des instances réelles.

Une autre application de la synchronisation est le marquage des routes. Pour peindre la signalisation sur les routes, des véhicules sont amenés à se déplacer tout en peignant. Lorsque le véhicule est vide, il doit se recharger en peinture à son dépôt d'origine. Salazar-Aguilar et al. [91] ont introduit des points de rechargement mobiles dans le réseau routier. Le problème nouveau est nommé Synchronized Arc and Node Routing Problem (SANRP). Sur ces points, un véhicule de type citerne vient recharger les véhicules affectés au marquage au sol. Une méthode de type ALNS est proposée pour résoudre le problème. Les auteurs ont montré que l'ajout de points de rechargement mobile dans le réseau apporte des gains sur la fonction objectif tant sur le nombre de kilomètres parcourus que sur le temps d'attente des véhicules.

Bredström et Rönnqvist [18] étendent le modèle de Vehicle Routing and Scheduling Problem with Time Windows (VRSP-TW) pour y ajouter des contraintes de précédences et de synchronisations. Une contrainte de synchronisation impose le service simultané de deux nœuds par deux véhicules. Une contrainte de précédence impose le début de service d'un nœud j après une durée S_{ij} du début de service du nœud i. La fonction objectif minimise une somme pondérée de trois termes : (i) la préférence entre un véhicule et un client, (ii) la somme des temps de trajets et (iii) l'équilibrage des tournées. Cette modélisation permet de traiter plusieurs cas réels en fonction de la valeur du paramètre S_{ij}. Pour résoudre ce problème, une heuristique résout séquentiellement des versions restreintes du problème où certaines affectations entre véhicules et clients ne sont pas autorisées. Dix instances sont générées et l'heuristique permet d'en résoudre la moitié à l'optimalité en une heure de calcul. La méthode montre toutefois des limites pour des instances plus grandes. Basée sur ces travaux, Afifi et al. [1] proposent une autre méthode heuristique s'appuyant sur le recuit simulé avec une méthode de recherche locale en utilisant des opérateurs courants (2-opt, or-opt, replacement, single-more). Leur méthode améliore en terme de qualité de la solution et de temps de calcul les résultats obtenus par Bredström et Rönnqvist [18]. Leurs temps de calculs sont de l'ordre de la seconde.

Récemment, l'heuristique de type VNS a été implémentée avec succès par Grangier et al. [52] pour résoudre les problèmes de transferts sur des plateformes Vehicle Routing Problem with Cross Docking (VRPCD), ainsi que la méthode ALNS pour le problème de tournées de véhicules à deux échelons avec synchronisations Two-Echelon Multiple-Trip Vehicle Routing Problem with Satellite Synchronization (2E-MTVRP-SS), Grangier et al. [53].

La problématique de synchronisation dans les problèmes de tournées de véhicules n'a été clairement identifiée que récemment dans la littérature, notamment par l'état de l'art de Drexl [35]. Les problèmes de transport et notamment de transport de personnes, deviennent de plus en plus interconnectées et synchronisées d'où l'émergence d'une littérature à ce sujet. Les problèmes qui en découlent restent au moins aussi difficiles à résoudre que les problèmes sans ces contraintes de synchronisation.

3.3.2 Contraintes temporelles

Un grand nombre des méthodes précédemment présentées sont des méthodes à base d'heuristiques ou de métaheuristiques. À la différence des méthodes de résolution exactes, ces méthodes s'appuient sur l'exploration de voisinages pour améliorer une solution initiale. Certains voisinages sont conçus par des permutations entre des clients au sein d'une tournée. Dans un problème de tournées de véhicules sans fenêtre de temps, pour ces voisinages, la permutation de deux clients ne changent pas la réalisabilité temporelle de la tournée concernée au sein de la solution. En revanche, pour le même problème avec des fenêtres de temps, une permutation de deux clients peut aboutir sur une solution irréalisable.

Pour tester de manière efficace la réalisabilité de voisinages, les *forward time slacks* ont été introduits par Savelsbergh [92, 93]. Pour tester si une permutation d'arcs dans une tournée est faisable, une variable indique la quantité de temps maximum de laquelle le début d'un service à un point peut être décalé. Au lieu de calculer l'ordonnancement des clients d'une tournée, cette méthode permet de tester en temps constant si la permutation est réalisable. Cette méthode a été testée avec succès sur les problèmes de TSP et VRP. Cordeau et al. [25] appellent cette variable : *forward time slack*.

Cette façon incrémentale, de tester la réalisabilité des solutions, fait le succès des méthodes heuristiques et métaheuristiques basées sur l'exploration des voisinages et sur les techniques d'insertion et de retrait.

Dernièrement, la méthode a été implémentée dans les algorithmes de type ALNS pour résoudre des problèmes de transports à la personne ([72]), des problèmes de tournées de véhicules à deux échelons avec synchronisation aux satellites ([52]). Dans ce dernier papier, les auteurs ont montré que l'utilisation des *forward time slacks* permet d'améliorer le temps de calcul d'une insertion de 90% par rapport à une méthode basée sur le calcul d'un diagramme PERT.

L'aspect temporel est devenu une contrainte facilement intégrable dans les méthodes approchées pour résoudre les problèmes de collectes et livraisons. La méthode basée sur les *forward time slacks* a permis de progresser en nombre d'explorations de solutions ce qui permet de traiter des problèmes plus rapidement ou bien de traiter des problèmes de plus grandes tailles.

3.4 Conclusion

La revue de la littérature nous amène à plusieurs conclusions. D'une part, les problèmes de tournées de véhicules avec collectes et livraisons sont difficiles à résoudre. Les meilleurs algorithmes exacts proposent des solutions optimales en une heure de calcul pour des problèmes à une centaine de clients. Les métaheuristiques, dont l'ALNS, offrent la possibilité de résoudre des instances plus larges dans des temps de calcul de l'ordre de la minute. La thématique de livraison des enrobés n'a pas reçu beaucoup d'attention dans la littérature. Le transport des bétons est un problème proche du nôtre avec certaines différences. L'élaboration de tournées utilisant une flotte hétérogène de véhicules ajoute de la difficulté à la résolution. Les contraintes de synchronisations complexifient également le problème. Pour ces dernières, les techniques de vérification de réalisabilité sont un point clé de la résolution.

Nous proposons deux tableaux pour synthétiser les informations recueillies dans cette revue de la littérature. Le premier tableau 3.1 présente les différentes caractéristiques des problèmes présentés ci dessus. Le second tableau 3.2 présente les composantes de la fonction objectif des mêmes problèmes.

Aucun problème recensé de la littérature ne combine simultanément toutes les contraintes de notre problème. Cette revue nous incite à développer une métaheuristique pour résoudre le problème industriel présenté dans cette thèse.

Référence	Année	Problème	**Demande**		**Chargement**		**Flotte**			**RS**	**TW**	Méthode	Application
			$< Q$	$> Q$	LTL	FTL	HVRP	F	D				
Rubasheuskaya [88]	2012	CSTP		•	•		•					-	AC
Brachner [17]	2013	VRPMS	•		•						•	R&R	AC
Durbin [39]	2003	-		•		•	•	•	•			DSS	RMC
Schmid [94]	2007	VRP*		•		•	•			•	•	[1]	RMC
Durbin et Hoffman [40]	2008	VRP		•		•	•	•	•		•	solveur	RMC
Schmid et al. [96]	2009	VRP*		•		•	•			•	•	VNS	RMC
Asbach et al. [8]	2009	VRPTW		•		•	•			•	•	LS	RMC
Schmid et al. [95]	2010	VRP*		•		•	•			•	•	VLNS	RMC
Liu et al. [69]	2014	VRPTW		•		•	•			•	•	GA	RMC
Weintraub et al. [104]	1996	VRPTW		•		•	•			•	•	H	WL
Palmgren et al. [77]	2001	LTSP		•		•					•	CG	WL
Palmgren et al. [78]	2004	LTSP		•		•					•	H	WL
Gronalt et Hirsch [55]	2007	TTVRP	•			•	•				•	TS	WL
Flisberg et al. [45]	2009	VRPTW		•		•	•				•	S	WL
Hirsch [57]	2011	TTVRP	•			•	•				•	TS	WL
El Hachemi et al. [44]	2013	LTSP		•		•				•	•	CP + ILS	WL
El Hachemi et al. [43]	2015	LTSP		•		•				•	•	S	WL
Arunapuram et al. [7]	2003	VRPFL				•					•	CG	-
Currie et Salhi [27]	2003	PDPTW	•			•	•	•	•		•	H	CONS
Gronalt et al. [54]	2003	FTPDPTW	•			•					•	H	-
Currie et Salhi [28]	2004	PDPTW	•			•	•	•	•		•	TS	CONS
Liu et al. [67]	2010	MDCARPFL		•		•						2P	-
Venkateshan et Mathur [103]	2011	VRPFL		•		•	•					CG	-
Salazar-Aguilar et al. [91]	2013	SANRP	•			•					•	H	SNOW
notre problème	2015	FT-PDP-RS		•		•	•	•	•	•	•	ALNS	AC

Tableau 3.1 – Ce tableau montre les différentes caractéristiques des références citées dans cette revue. Les colonnes présentent le type de problème, la demande ($< Q$ si la demande est inférieure à la capacité des véhicules, $> Q$ le cas contraire), le chargement (LTL pour *Less Than Truckload* et FTL pour *Full TruckLoad*), la flotte (HVRP pour flotte hétérogène, F pour coût fixe et D pour coût de trajets dépendants des véhicules), la synchronisation sur les ressources (RS), la présente de fenêtre de temps (TW) et enfin la méthode de résolution et le domaine d'application. Les abbréviations de la colonne méthode de résolution sont : R&R (Ruin and Recreate), DSS (Discret Simulation System), MCNF (Multi-Commodity Flow Network), VNS (Variable Neighborhood Search), LS (Local Search), VLNS (Variable Large Neighborhood Search), GA (Genetic Algorithm), H (Heuristic), CG (Column Generation), S (Solver), TS (Tabu Search), CP (Constraint Programming), ILS (Iterative Local Search), 2P (2-phase). Les applications des problèmes sont les suivantes : AC (Enrobés), RMC (Bétons), WL (Transport de bois), CONS (Matériaux de construction), SNOW (marquage au sol).

Référence	Année	Problème	**Véhicules**				**Demandes**		
			F	T	Op	WT	E	D	NS
Rubasheuskaya [88]	2012	CSTP		•	•	•	•		
Brachner [17]	2013	VRPMS		•	•				
Durbin [39]	2003			•					
Schmid [94]	2007	VRP⋆		•				•	
Durbin et Hoffman [40]	2008	VRP		•					
Schmid et al. [96]	2009	VRP⋆		•				•	
Asbach et al. [8]	2009	VRPTW		•					•
Schmid et al. [95]	2010	VRP⋆		•				•	
Liu et al. [69]	2014	VRPTW		•				•	
Weintraub et al. [104]	1996	VRPTW		•					
Palmgren et al. [77]	2001	LTSP		•					
Palmgren et al. [78]	2004	LTSP		•					
Gronalt et Hirsch [55]	2007	TTVRP		•					
Flisberg et al. [45]	2009	VRPTW		•	•				•
Hirsch [57]	2011	TTVRP		•					
El Hachemi et al. [44]	2013	LTSP		•		•			
El Hachemi et al. [43]	2015	LTSP		•		•			
Arunapuram et al. [7]	2003	VRPFL		•					
Currie et Salhi [27]	2003	PDPTW	•	•					
Gronalt et al. [54]	2003	FTPDPTW		•	•				
Currie et Salhi [28]	2004	PDPTW	•	•					
Liu et al. [67]	2010	MDCARPFL		•					
Venkateshan et Mathur [103]	2011	VRPFL		•					
Salazar-Aguilar et al. [91]	2013	SANRP		•					
notre problème	2015	FT-PDPT-TW	•	•	•	•			•

Tableau 3.2 – Ce tableau présente les coûts considérés composant les fonctions objectifs rencontrées dans la littérature (problème de minimisation). Pour la partie **Véhicules**, les coûts suivants peuvent être pris en compte : F (coût fixe d'utilisation d'un véhicule), T (coût dépendant de la distance), Op (coût dépendant du temps mis pour exécuter l'opération de chargement ou déchargement), WT (le coût lié à l'attente aux points). Pour la colonne des **Demandes**, sont pris en compte E (le coût lié à l'heure de fin de réalisation d'une demande), D (le coût lié aux retards d'une demande) et NS (le coût lié à la non satisfaction d'une demande).

Méthodologie de résolution et décomposition du problème

Table des matières

Cette partie traite de la méthodologie de résolution retenue pour le problème industriel. Celui-ci est un problème opérationnel comportant des demandes de transport en matériaux définies sur un horizon hebdomadaire. La gestion des opérations étant effectuée sur un horizon journalier, nous proposons une méthode pour découper les demandes de transports hebdomadaires en demandes de transports journalières. Cette démarche de décomposition est présentée dans le chapitre 4.

Notre problème est alors scindé en deux phases :

- une phase de découpage des demandes, notée $\mathcal{P}_1$;
- une phase de création des tournées, notée $\mathcal{P}_2$.

La phase de découpage $\mathcal{P}_1$ est traitée dans le chapitre 5 de cette partie. Ensuite, le chapitre 6 détaille la méthode retenue pour la création des requêtes.

La phase $\mathcal{P}_2$ fait, quant à elle, l'objet d'une étude particulière traitée dans la partie III.

4

Décomposition du problème

Sommaire

Dans ce chapitre, nous présentons la méthodologie retenue pour traiter le problème présenté dans la section 2.1. Suite à l'analyse de la revue de la littérature, nous orientons nos travaux vers une résolution approchée du problème. Ce choix est motivé par deux raisons principales. La première raison est la complexité du problème à traiter. Les algorithmes dits exacts ne permettent pas, aujourd'hui, de résoudre des instances de taille importante dans des temps de résolution corrects vis à vis des exigences industrielles. La seconde raison est la bonne qualité des solutions fournies par les algorithmes approchés de la littérature sur les problèmes traitant des tournées de véhicules. L'ALNS, présenté par Ropke et Pisinger [87], montre de bonnes solutions pour le problème de PDPTW [87] et les problèmes de tournées avec synchronisation [72, 52].

Pour résoudre le problème, nous proposons une méthode en deux phases. Les motivations de ce choix sont présentées en section 4.1. Nous décrivons l'approche métier existante dans l'entreprise dans la section 4.2. La méthode en deux phases est présentée dans la section 4.3. Enfin, dans la section 4.4, nous présentons les différents modes de fonctionnement de notre méthode.

4.1 Motivations

Le problème complet présenté dans la section 2.1 comporte deux problématiques identifiées et propres aux tournées de véhicules.

La quantité de produits devant être livrée pour chaque demande excède la capacité nominale des camions. Plusieurs camions sont nécessaires pour satisfaire une demande. Nous devons donc découper ces demandes en requêtes. La première problématique porte sur le découpage des demandes en requêtes et sur une pré-affectation de ces requêtes à une catégorie de véhicules en fonction de la flotte disponible. Cette question sera abordée par la première phase du problème dite, phase de découpage.

La deuxième problématique porte sur la construction des tournées. Pour chaque requête, nous devons choisir le véhicule qui la livrera et la position de la requête dans la tournée. Cette phase d'élaboration des

tournées correspond à la deuxième phase du problème.

Différentes méthodes ont été proposées dans la littérature pour traiter des problèmes similaires au nôtre. Ebben et al. [41] présentent une méthode d'ordonnancement à base de règles pour traiter un problème de véhicules à guidage automatique dans un aéroport (AGV). Les requêtes sont ordonnées et sélectionnées suivant des règles pour construire un ordonnancement. Plusieurs ordonnancements peuvent être construits en fonction des règles choisies et celui de meilleur coût en fonction d'un critère est sélectionné. Les critères possibles sont la minimisation du nombre de requêtes en retard, la minimisation de la somme des retards, la minimisation des trajets à vide et la maximisation des requêtes servies en avance. Les règles choisies permettent d'obtenir un taux de service supérieur à 90% dans la majorité des instances résolues. Cependant, cette méthode ne traite pas de l'utilisation d'une flotte de capacité hétérogène. C'est une des perspectives de leur travail.

Une méthode hybride est proposée par Schmid et al. [95] combinant un algorithme VLNS et un solveur MILP. Le problème résolu est la livraison de bétons pour la construction. La contrainte de continuité dans le service est pénalisée dans la fonction objectif. Même si la méthode est originale et présente de bonnes perspectives, elle résout des instances de tailles moyennes (environ 50 demandes) sous une heure de calcul avec un gap de 12% à la borne inférieure. Seulement le nœud origine de l'arbre de recherche peut être résolu dans le temps imparti. Pour découper les demandes en requêtes, les auteurs créent un nombre de requêtes pour chaque demande supérieur au besoin réel. Seulement un sous-ensemble de requêtes satisfaisant la quantité de produit nécessaire est visité. Cela contribue à complexifier le problème pour le solveur MILP.

Les mêmes auteurs ont proposé une discrétisation du temps pour le même problème [96]. Cette méthode n'a pas présenté de meilleurs résultats. Le problème résolu est similaire au nôtre et les auteurs soulignent la difficulté à trouver de très bonnes solutions. Dans leur modélisation, il n'y a pas mention de synchronisations aux ressources mais plutôt de continuité de service.

La méthode de découpage des demandes en requêtes en proposant toutes les combinaisons a également été utilisée par Asbach [8]. Leur modélisation inclut un temps minimal (*minimum time lag*) et un temps maximal (*maximum time lag*) entre deux requêtes pour empêcher le béton de figer entre deux livraisons successives. Une résolution avec le solveur MIP CPLEX a été essayée mais n'a pas retourné de solution optimale. Une méthode heuristique à base de recherche locale est proposée pour résoudre le problème.

La littérature étudiée propose plusieurs méthodes pour résoudre des problèmes très proches du notre. Le découpage des demandes en requêtes avec toutes les combinaisons possibles en fonction des catégories de véhicules a l'inconvénient de proposer un modèle trop conséquent et ne peut pas être résolu par des solveurs MIP ou MILP. La littérature s'oriente vers des méthodes heuristiques à base de voisinage pour résoudre ces problèmes.

De plus, aucun problème ne considère la synchronisation aux ressources. La synchronisation dans les problèmes de tournées fait l'objet d'une attention récente de la communauté scientifique [35]. Une méthode ALNS a déjà été implémentée avec succès pour résoudre le DARPT [72] qui traite de la synchronisation entre des véhicules pour le transport de personnes.

Cette utilisation fructueuse de l'ALNS a motivé son utilisation pour notre problème dans lequel les contraintes de synchronisation sont une contribution scientifique au regard de la littérature.

4.2 Approche métier

Dans cette section, nous décrivons la pratique actuelle de l'entreprise pour le service des demandes. Un responsable logistique planifie quotidiennement les tournées des camions en fonction des demandes en matériaux des chantiers de l'entreprise.

Il définit des rotations pour chaque camion, c'est à dire un aller/retour depuis un site de collecte (chargement du camion) vers un site de livraison (déchargement du camion). Le responsable logistique définit des rotations servant une seule demande, le retour du point de livraison au point de collecte s'effectue donc à vide. Si la proximité géographique de deux demandes est justifiée ou bien si le point de livraison d'une

demande 1 coïncide avec le point de livraison d'une demande 2, alors, il peut choisir de mutualiser deux demandes au sein d'une même rotation d'un camion.

Cette planification concerne uniquement les flux tendus (enrobés) pour lesquels la contrainte de continuité dans le service impose un cadencement précis des véhicules. Les flux détendus ne sont pas précisément planifiés par le responsable logistique. Des camions non utilisés pour les flux tendus sont mis à disposition des chantiers ou des plateformes pour gérer l'approvisionnement et les évacuations.

Ainsi, pour chaque journée de travail, le responsable logistique fournit une feuille de route aux camions servant les flux tendus. À l'issue de leur service, après un échange entre le chauffeur et le responsable logistique, certains camions peuvent se voir affecter d'autres tâches de transport.

L'optique de ce travail de recherche est de fournir une planification centralisée des tournées de camions. La méthode actuelle montre ses limites qui sont :

- ordonnancement non précis : hormis les camions traitant les flux tendus, les autres véhicules n'ont pas de feuille de route précise. Chaque camion est libre de son parcours et de son trajet. Les chauffeurs ont connaissance de l'état du réseau et jugent, par leur expérience, du trajet à parcourir.
- sous-exploitation des moyens : certains camions qui ont fini leurs transports en flux tendus peuvent finir leur journée alors qu'ils auraient pu servir d'autres demandes de transport.
- l'ordonnancement ne tient pas compte des contraintes de synchronisation aux ressources : deux camions peuvent se trouver sur le même site et vouloir faire un chargement ou un déchargement en même temps.
- irréalisabilité du planning : les feuilles de routes ne tiennent pas compte précisément des temps de trajet. Certains camions peuvent arriver trop tôt ou trop tard sur leurs lieux de service.

Notre travail a pour objectif de pallier à ces limites en proposant un planning de feuilles de route servant toutes les requêtes. Nous nous attacherons à optimiser les moyens disponibles tout en produisant un ordonnancement réalisable.

4.3 Méthodologie retenue

Pour satisfaire les objectifs précédemment cités, nous proposons une méthode en deux phases.

La première phase de découpage des demandes en requêtes est notée $\mathcal{P}_1$. Par rapport aux modélisations de la littérature où le découpage n'affecte pas les requêtes à des camions, nous choisissons de découper de manière exacte les demandes en fonction des catégories de véhicules disponibles.

Ce découpage est modélisé sous forme d'un Programme Linéaire (PL). Il prend en compte les caractéristiques intrinsèques des catégories de camions ainsi que les quantités de produits à transporter. Ce découpage fournit en sortie le nombre de requêtes à effectuer pour chaque demande et chaque catégorie de camions. Chaque requête correspond au transport partiel du matériau d'une demande à hauteur de la capacité du camion de la catégorie duquel il est issu.

Ce découpage n'explore pas toutes les possibilités d'affectation. En effet, une requête créée de $25t$ ne pourra pas être servie par un camion dont la capacité de transport est de $12.5t$.

En revanche, ce découpage par sa modélisation garantit la réalisabilité du service des demandes. L'ensemble des demandes devra pouvoir être servi par les camions. De plus, ce modèle offre la possibilité à deux demandes proches de pouvoir être servies au sein d'une même rotation par un même véhicule et le même jour.

Le découpage permet de fournir des solutions optimales qui ne sont pas forcément intuitives. Considérons une demande n_1 d'un produit où la quantité $q_{n_1} = 30t$ doit être livrée entre un point de collecte et un point de livraison distant de 10km. La flotte de véhicules est composée d'un véhicule k_1 de capacité

Véhicule	Capacité t	Coût fixe d'utilisation €	Coût kilométrique €/km	Nombre de rotations	Coût total €
k_1	25	141	0.65	2	206
k_2	12	180	0.35	3	162

Tableau 4.1 – Exemple de l'utilisation de véhicules pour une demande. Le tableau présente la capacité des véhicules c (en t), le coût journalier $\mathcal{CH}$ (en €), le coût kilométrique $\mathcal{CK}$ (en €/km), le nombre de rotations et le coût total.

$c_{k_1} = 25t$ et d'un véhicule k_2 de capacité $c_{k_2} = 12t$. Les coûts journaliers $\mathcal{CH}$ et coûts kilométriques $\mathcal{CK}$ d'utilisation des véhicules sont présentés dans le tableau 4.1.

Pour satisfaire cette demande et en considérant un retour au point de collecte, le véhicule k_1 devra effectuer 2 transports (aller-retour) entre le point de collecte et le point de livraison tandis que le véhicule k_2 effectuera 3 transports. En minimisant le nombre de tournées, le véhicule k_1 présente un meilleur potentiel puisqu'il effectuera un transport de moins que le véhicule k_2. En revanche, si nous souhaitons minimiser les coûts, ce choix se révèle moins bon (206€ contre 162€).

Le problème de découpage de la première phase est résolu de manière exacte par un solveur MIP. La deuxième phase de la méthode de résolution est notée $\mathcal{P}_2$. Elle correspond à l'élaboration des tournées des camions. Pour cela, nous utilisons l'algorithme ALNS.

Nous proposons une schématisation des méthodes de résolution retenues dans la figure 4.1.

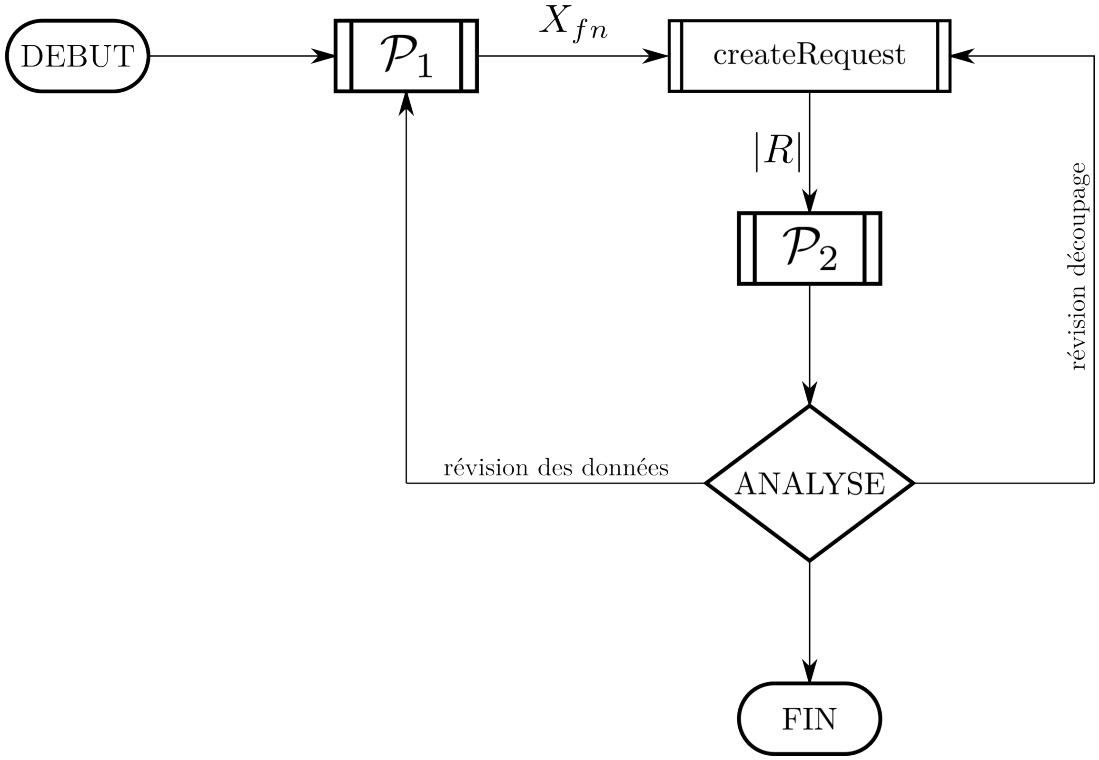


FIGURE 4.1 – Algorigramme de notre méthode.

Le fonctionnement de notre algorithme dans sa totalité est le suivant :

1. DEBUT : lecture des données d'entrée ;
2. $\mathcal{P}_1$: découpage des demandes (variables X_{fn}) ;
3. createRequest : création des requêtes de transport (ensemble R) ;

4. $\mathcal{P}_2$: création des tournées de véhicules ;

5. ANALYSE : analyse de la solution retournée. Possibilité de réviser les données d'entrée. Possibilité de remettre en cause l'affectation ;

6. FIN : restitution de la solution finale.

Notre algorithme propose deux possibilités de *backtracking* en analysant la solution finale retenue. Ces fonctionnalités permettent de modifier le découpage de la phase $\mathcal{P}_1$ si une solution n'est pas jugée satisfaisante au regard des critères d'analyse.

4.4 Fonctionnement

Le fonctionnement réel de notre algorithme s'appuie sur sa description présentée dans la section 4.3.

Les données d'entrée de l'entreprise sont un ensemble de demandes de transport en matériaux ayant jusqu'à une semaine comme amplitude horaire.

Dans le cadre de notre étude, nous pouvons utiliser la phase $\mathcal{P}_1$ de deux façons différentes :

1. Utilisation monopériodique : la phase $\mathcal{P}_1$ est utilisée pour faire le découpage de demandes d'une journée donnée en fonction des véhicules disponibles sur cette même journée.

2. Utilisation multipériodique : la phase $\mathcal{P}_1$ est utilisée pour faire le découpage d'un ensemble de demandes, pouvant s'étaler sur plusieurs périodes et en fonction des véhicules disponibles sur cette période.

Dans une utilisation industrielle, l'utilisation monopériodique correspond à une journée de travail, l'utilisation multipériodique à une utilisation sur une semaine de travail.

La phase $\mathcal{P}_2$ fonctionne pour une période donnée. Elle se charge d'affecter les requêtes créées (variable X_{fn}) à des véhicules. Dans le cas d'une utilisation multipériodique, la phase $\mathcal{P}_2$ est appelée pour chaque période de l'horizon temporel.

La réalisabilité d'une solution est étudiée lors de la phase $\mathcal{P}_2$. Suivant l'issue de cette analyse, il peut y avoir une révision de l'algorithme :

- Utilisation monopériodique : des requêtes n'ont pas pu être intégrées. Un nouveau découpage peut être proposé pour affecter d'une autre manière les requêtes aux véhicules.

- Utilisation multipériodique : des requêtes n'ont pas pu être intégrées lors de certaines périodes. Ces requêtes peuvent être déplacées sur une période adjacente (précédente ou suivante). Si cette procédure se révèle infructueuse, alors une révision des données d'entrée peut être effectuée.

Dans le cas d'un fonctionnement réel, certaines requêtes peuvent être créées à la volée et être intégrées dans le fonctionnement de la phase $\mathcal{P}_2$. Cette fonctionnalité peut être implémentée aisément dans notre algorithme mais ne fait pas l'objet de l'étude de cette thèse. Elle couvre les champs de recherche du Dynamic Vehicle Routing Problem notamment présentés dans les travaux de thèse de Pillac [82].

5

Phase $\mathcal{P}_1$: Découpage des demandes

Sommaire

Ce chapitre est consacré à l'étude de la phase $\mathcal{P}_1$. Les enjeux et la description des éléments importants sont présentés dans la section 5.1. Le modèle de résolution monopériodique est présenté dans la section 5.2 et le modèle de résolution multipériodique dans la section 5.3.

5.1 Description et enjeux de la phase $\mathcal{P}_1$

La phase $\mathcal{P}_1$ permet le découpage des demandes en requêtes. Cette étape constitue un choix scientifique pour nos travaux par rapport à d'autres possibilités de la littérature.

La description du problème $\mathcal{P}_1$ est la suivante. Chaque demande de transport doit être servie par des camions livrant chacun une partie de la quantité, une requête, à transporter de cette demande. Les camions ont une disponibilité temporelle. La création d'une requête a un coût. Nous cherchons à minimiser le coût total de découpage des demandes en requêtes.

La disponibilité des véhicules est assimilée à une ressource temporelle. Ce problème d'affectation est inspiré du problème de transport (Transportation Problem), lui même dérivé du problème de flots à coût minimum (Minimum Cost Flow Problem). Celui-ci a été initialement introduit par Hitchcock [58].

Dans le problème de transport, il faut minimiser le coût d'affectation entre des produits et des véhicules sous les contraintes de satisfaction de la capacité (contrainte de capacité) et de consommation de ressource (contrainte temporelle). La modélisation de notre problème $\mathcal{P}_1$ est très proche du problème de transport.

Les données d'entrée de $\mathcal{P}_1$ sont les demandes ainsi que les véhicules disponibles pour une période donnée. En sortie, l'algorithme indique la quantité de requêtes à créer pour chaque demande et chaque catégorie de véhicules. À ce stade, nous introduisons la notion de rotation représentant la rotation d'un véhicule servant une demande. Ce concept est expliqué plus précisément dans la section 5.1.3.

Le problème $\mathcal{P}_1$ est défini sur un graphe complet $G' = (V', A')$. L'ensemble des sommets est V' et l'ensemble des arcs est A'. L'ensemble des demandes est noté N et l'ensemble des catégories de véhicules est noté F. Le temps de trajet entre deux sommets $(i, j) \in V^2$ et pour un véhicule appartenant à la catégorie $f \in F$ est noté t_{ij}^f. L'ensemble des rotations possibles est noté Ω. Le coût d'affectation d'une rotation $\omega \in \Omega$ est noté C_ω. L'ensemble des véhicules pouvant servir la demande $n \in N$ est noté $F_n \subset F$. Par analogie, l'ensemble des demandes pouvant être servies par un véhicule de la catégorie $f \in F$ est noté $N_f \subset N$.

Dans la suite de cette section, nous décrivons en détail les demandes, les catégories de véhicules ainsi que la notion de rotation.

5.1.1 Description des demandes

Les demandes constituent les données d'entrée de la phase $\mathcal{P}_1$. Le tableau 5.1 détaille les différents attributs d'une demande n.

Attribut	Valeur
p_n	point de livraison
d_n	point de collecte
e_{p_n}	heure d'ouverture du service de la collecte
l_{p_n}	heure de fermeture du service de la collecte
e_{d_n}	heure d'ouverture du service de la livraison
l_{d_n}	heure de fermeture du service de la livraison
q_n	quantité de matière à transporter

Tableau 5.1 – Attributs d'une classe Demand.

Chaque demande est composée d'un point de collecte $p_n \in P' \subset V$ et d'un point de livraison $d_n \in D' \subset V$.

5.1.2 Description des véhicules

La flotte de véhicules est modélisée comme une ressource temporelle. Pour réduire le nombre de variables du problème $\mathcal{P}_1$, nous considérons des catégories de véhicules, regroupant plusieurs véhicules partageant les mêmes propriétés et non des véhicules individuellement. L'ensemble des catégories de véhicules est noté F. Les $nbVehicle_f$ véhicules appartenant à une même catégorie $f \in F$ partagent les attributs suivants :

- capacité en t d'un véhicule Q^f ;
- coût kilométrique en €/km d'utilisation d'un véhicule $\mathcal{CK}^f$;
- coût horaire en €/h d'utilisation d'un véhicule $\mathcal{CH}^f$;
- coût fixe $\mathcal{CD}^f$ en € d'utilisation d'un véhicule mais qui n'est pas pris en compte dans le cadre de cette modélisation.

L'ensemble des dépôts de tous les véhicules est noté $O \subset V$.

Pour chaque période du problème, les véhicules disposent d'une durée de travail H. Sur cet horizon de temps, les véhicules effectuent des trajets liés à la livraison des produits entre des points de collecte et des points de livraison. Ils effectuent également un trajet depuis le dépôt d'origine en début de période ainsi qu'un retour au dépôt d'arrivée en fin de période.

Lors de la phase $\mathcal{P}_1$, nous n'avons pas la connaissance exacte du trajet réalisé par le véhicule depuis son dépôt d'origine et vers son dépôt d'arrivée dans la solution finale. En première estimation, le temps *disponible* pour la réalisation des trajets est calculé en déduisant, de la durée de la journée de travail, le temps de trajet au sommet de collecte le plus proche et le temps depuis le sommet de livraison le plus proche du dépôt. Nous définissons par $t^{\min}_{pickup}$ (respectivement $t^{\min}_{delivery}$) le temps de trajet minimum entre le dépôt d'origine et les points de collecte (resp. les points de livraison).

$$t^{\min}_{pickup} = \min_{\forall o \in O, \forall i \in P', o \neq i} t_{op_i}$$

$$t^{\min}_{delivery} = \min_{\forall i \in D', \forall o \in O, o \neq i} t_{d_i o}$$

La durée de travail corrigé $\bar{H}$ pour chaque période est ainsi définie par :

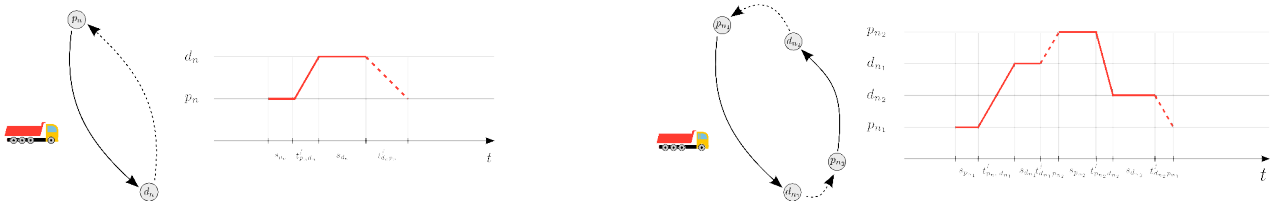
$$\bar{H} = H - t^{\min}_{pickup} - t^{\min}_{delivery}$$

5.1.3 Notion de rotation

Pour réaliser ce découpage et estimer l'ensemble des requêtes R, préalablement à la construction des tournées, on définit la notion de rotation (ensemble Ω). Une rotation est composée d'un aller, camion chargé en matériau, représentant le service d'une requête et d'un retour à vide. Cette notion est issue de la pratique métier, section 4.2, utilisée dans l'entreprise par le responsable logistique. C'est le nombre de rotations nécessaires pour chaque type de véhicule que nous allons chercher à définir lors de la phase $\mathcal{P}_1$.

Dans un premier temps, cette notion considère uniquement un aller-retour d'une demande n mais ne prend pas en compte le trajet depuis le dépôt d'origine et vers le dépôt d'arrivée. Ce concept est ensuite étendu à l'enchainement de deux demandes n_1 et n_2. Chaque rotation consomme une certaine quantité de ressource temporelle et livre une certaine quantité de matériaux.

Nous identifions ces différentes rotations pouvant servir une demande ou plus par la notation Ω_i lorsqu'une rotation visite i demandes différentes. Dans le cadre de cette étude, nous nous limiterons à la visite de deux clients maximum $i \leq 2$. La figure 5.1 présente un schéma d'une rotation à une demande (5.1a) et d'une rotation à deux demandes (5.1b).



(a) Une rotation simple de type Ω_1 (b) Une rotation composée de type Ω_2

FIGURE 5.1 – Représentation schématique des différents types de rotations.

L'ensemble des rotations est $\Omega = \Omega_1 \cup \Omega_2$. Une rotation $\omega_1 \in \Omega_1$ lie une demande n avec une catégorie de véhicules f. Indifféremment, le coût de cette rotation peut être écrit C_{fn} ou C_{ω_1} ainsi que la durée écoulée de cette rotation qui peut être écrit T_{fn} ou T_{ω_1}. Idem, pour une rotation $\omega_2 \in \Omega_2$ liant les demandes n_1 et n_2 avec une catégorie de véhicules f. Le coût peut être écrit $C_{fn_1n_2}$ ou C_{ω_2} et la durée $T_{fn_1n_2}$ ou T_{ω_2}.

Nous introduisons plusieurs sous-ensembles de Ω. Nous les notons sous la forme $\Omega^{\alpha_1/\alpha_2/\alpha_3}$ avec $\alpha_1 = f$ le paramètre qui désigne le sous-ensemble de rotations pouvant être effectuées par un véhicule de la

catégorie f, $\alpha_2 = n$ le paramètre qui désigne le sous-ensemble de rotations pouvant servir la demande n et $\alpha_3 = t$ le paramètre qui désigne le sous-ensemble de rotations pouvant être effectuées à la période t. Ces paramètres peuvent être associés mutuellement. Par exemple, le sous-ensemble $\Omega^{f//t}$ désigne le sous-ensemble des rotations effectuées par un véhicule de la catégorie f à la période t.

Description d'une rotation simple Ω_1

Une rotation de type Ω_1 est définie par le service partiel d'une et une seule demande n par un véhicule donné de la catégorie f. Le véhicule effectue son service, d'une durée s_{p_n}, au point de collecte p_n, se déplace jusqu'au point de livraison d_n, effectue son service, d'une durée s_{d_n}, au point de livraison et retourne au point de collecte p_n. La figure 5.2 présente cette rotation.

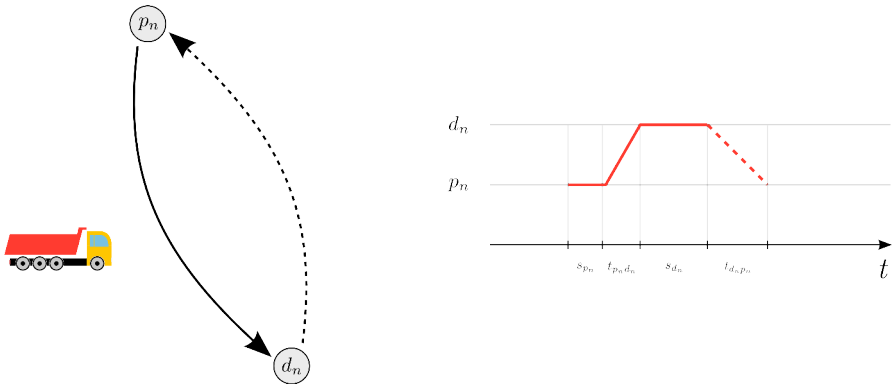


FIGURE 5.2 – Représentations d'une rotation de type Ω_1 entre un véhicule de la catégorie f et une demande n. Sur le dessin de gauche, nous représentons une illustration du graphe. Sur le dessin de droite, nous représentons le graphique temporel de cette rotation.

La durée nécessaire pour la réalisation de cette rotation est :

$$T_{fn} = s_{p_n} + t^f_{p_n d_n} + s_{d_n} + t^f_{d_n p_n}$$

Le coût C_{fn} de cette rotation est le coût lié aux trajets ainsi que le coût du temps passé sur les points de collecte et de livraison. Nous n'implémentons pas le coût fixe lié à l'utilisation du véhicule. Le coût est donné par la formule suivante :

$$C_{fn} = \mathcal{CH}^f * T_{fn} + \mathcal{CK}^f_{p_n d_n} + \mathcal{CK}^f_{d_n p_n}$$

Description d'une rotation composée Ω_2

Comme signalé en introduction, un des enjeux du découpage porte sur les aspects d'optimisation : le découpage doit permettre d'affecter deux demandes qui, si elles sont réalisées séquentiellement par le même véhicule, permettent une réduction de coût. C'est une des pratiques utilisées par le responsable logistique. Dans le cas de flux détendus, ces demandes doivent si possible être planifiées dans la même période. On définit cette rotation de la manière suivante.

Dans le cas d'une rotation de deux demandes séquentielles n_1 puis n_2, le trajet réalisé par le véhicule est le suivant : service au point de collecte de p_{n_1}, trajet du point de collecte vers le point de livraison de la demande n_1, service au point de livraison d_{n_1}, trajet du point de livraison de la demande n_1 vers le point de collecte de la demande n_2, service au point de collecte p_{n_2}, trajet du point de collecte vers le point de livraison de la demande d_{n_2}, service au point de livraison d_2 et retour vers le point de collecte de la demande 1. Cette rotation est illustrée par la figure 5.3.

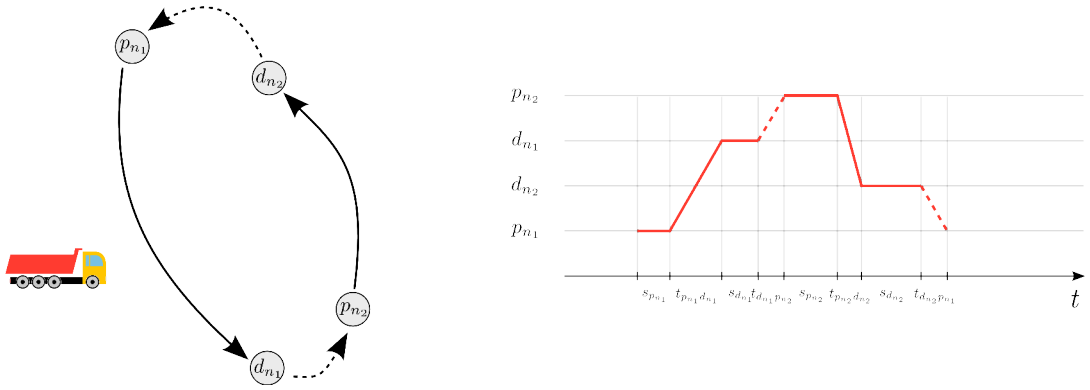


FIGURE 5.3 – Représentations d'une rotation de type Ω_2 entre un véhicule de la catégorie f et deux demandes n_1 et n_2. Sur le dessin de gauche, nous représentons une illustration du graphe. Sur le dessin de droite, nous représentons le graphique temporel de cette rotation avec le service de la demande n_1 puis le service de la demande n_2.

Cette rotation présente un potentiel d'optimisation uniquement s'il est bénéfique au regard du coût de faire une mutualisation des deux demandes. En posant $\mathcal{CKH}^f$ le coût horaire et kilométrique entre un point i et un point j servis par un véhicule de la catégorie f, ceci se traduit par la validité de l'inégalité suivante :

$$\mathcal{CKH}^f_{d_{n_1}p_{n_2}} + \mathcal{CKH}^f_{d_{n_2}p_{n_1}} < \mathcal{CKH}^f_{d_{n_1}p_{n_1}} + \mathcal{CKH}^f_{d_{n_2}p_{n_2}}$$

La durée nécessaire pour la réalisation de cette rotation est alors :

$$T_{fn_1n_2} = s_{p_{n_1}} + t^f_{p_{n_1}d_{n_1}} + s_{d_{n_1}} + t^f_{d_{n_1}p_{n_2}} + s_{p_{n_2}} + t^f_{p_{n_2}d_{n_2}} + s_{d_{n_2}} + t^f_{d_{n_2}p_{n_1}}$$

Le coût $C_{fn_1n_2}$ de cette rotation est le coût lié aux trajets ainsi que le coût du temps passé sur les points de collectes et de livraisons des deux demandes. Le coût est donné par la formule suivante :

$$C_{fn_1n_2} = \mathcal{CH}^f * (T_{fn_1n_2}) + \mathcal{CK}^f_{p_{n_1}d_{n_1}} + \mathcal{CK}^f_{d_{n_1}p_{n_2}} + \mathcal{CK}^f_{p_{n_2}d_{n_2}} + \mathcal{CK}^f_{d_{n_2}p_{n_1}}$$

Une telle représentation du découpage estime la consommation des ressources temporelles définie par la disponibilité de la flotte de véhicules. Cette approximation permet de majorer cette consommation de ressources et de garantir la réalisabilité du découpage en vue de la phase $\mathcal{P}_2$.

5.1.4 Enjeux

L'enjeu primordial de la phase $\mathcal{P}_1$ est de garantir la création d'un ensemble de requêtes R pouvant être insérées dans les tournées de véhicules lors de la phase $\mathcal{P}_2$.

En majorant le temps passé par les rotations Ω_1 ou Ω_2, nous sur-évaluons le temps d'occupation des véhicules. La solution retournée par la phase $\mathcal{P}_1$ est une estimation de l'occupation des véhicules qui majore, en terme de temps, la solution finale. Cette majoration permet de garantir la réalisabilité de la phase $\mathcal{P}_2$ lorsque nous devrons insérer les requêtes dans les tournées de véhicules.

Ceci est très important pour notre algorithme car nous préférons fournir une solution incomplète en sortie de phase $\mathcal{P}_2$ par rapport à aucune solution en sortie de phase $\mathcal{P}_1$ alors qu'une solution pourrait exister.

5.2 Modèle monopériodique

Le modèle de découpage monopériodique a pour objectif de découper les demandes de produits afin de créer des requêtes pour une flotte de véhicules lorsque l'on considère une période unique (typiquement une journée). Nous présentons deux modèles :

1. modèle avec des rotations simples (Ω_1), noté $\mathcal{P}_1(\Omega_1)$;
2. modèle avec des rotations composées (Ω_2), noté $\mathcal{P}_1(\Omega_{1,2})$.

5.2.1 Modèle monopériodique avec rotations simples

Dans ce modèle, uniquement les rotations Ω_1 sont considérées. Les notations utilisées sont celles de la section 5.1. Une rotation $\omega_1 \in \Omega_1$ représente un service d'une demande n par un véhicule de catégorie f à hauteur de la capacité de ce véhicule.

Nous introduisons les variables entières $X_{fn} \in \mathbb{N}$ représentant le nombre de rotation affectées à un véhicule de la catégorie $f \in F$ et une demande $n \in N$. Chaque rotation se traduira par la suite par une requête liant la demande n et la catégorie f et servant la quantité Q_f de matériau.

La modélisation mathématique du problème $\mathcal{P}_1(\Omega_1)$ est la suivante :

$$\mathcal{P}_1(\Omega_1)$$

$$\min \quad \sum_{f\in F}\sum_{n\in N} C_{fn}X_{fn} \tag{5.1}$$

$$s.c.$$

$$\sum_{n\in N_f} T_{fn}X_{fn} \leq \bar{H} * nbVehicle_f \qquad \forall f \in F \tag{5.2}$$

$$\sum_{f\in F_n} Q_f X_{fn} \geq q_n \qquad \forall n \in N \tag{5.3}$$

$$X_{fn} \in \mathbb{N}^+ \tag{5.4}$$

La fonction objectif du problème (5.1) est de minimiser le coût des rotations sélectionnées entre les véhicules et les demandes. Les contraintes (5.2) limitent le temps total d'utilisation des véhicules pour chaque catégorie de véhicules. Ce temps total est égal à la durée de travail corrigée par le nombre de véhicules dans la catégorie concernée. Les contraintes (5.3) garantissent que la quantité de produits livrée pour une demande par toutes les catégories de véhicules est respectée.

La résolution de ce modèle permet de créer un ensemble de requêtes de transport permettant de satisfaire les demandes. La variable X_{fn} indique le nombre de requêtes à créer servant la demande n et pouvant être faites par un véhicule de la catégorie f.

5.2.2 Ajout de bornes sur les variables X_{fn}

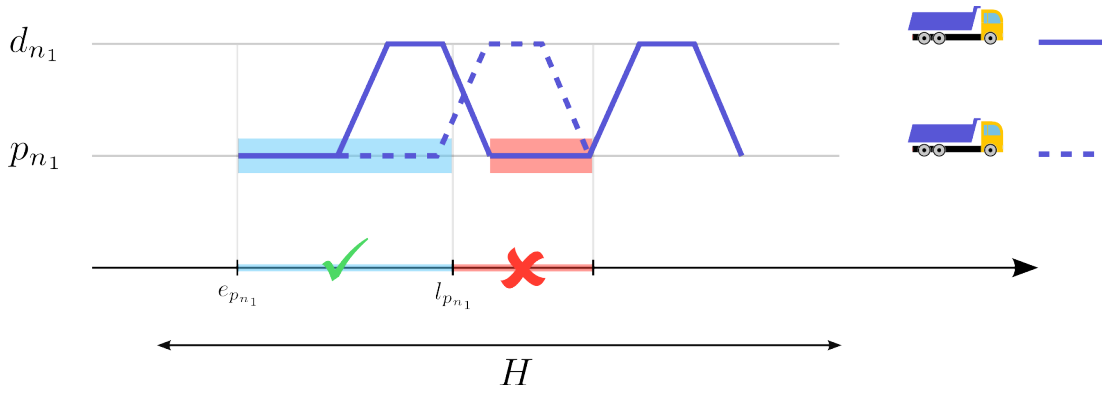
Les requêtes créées à partir des rotations se voient définir des fenêtres de temps calculées en fonction des fenêtres de temps de la demande dont elles sont issues. Cette partie est décrite précisément dans le chapitre 6.

Le modèle $\mathcal{P}_1(\Omega_1)$ précédent présente une limite importante liée à la présence de fenêtre de temps sur certains types de demande et relative à la solution retournée. En effet, les contraintes (5.2) limitent le domaine des variables X_{fn} par rapport à une estimation totale du temps disponible pour les véhicules d'une catégorie sur une période entière.

Cependant, certaines demandes à flux tendus ont des fenêtres de livraison qui ne s'étalent pas sur toute la période (ou toute la journée) mais plutôt sur une période plus limitée. Dans le cas d'une demande à satisfaire sur une fenêtre de temps étroite, il peut être nécessaire de faire appel à plusieurs véhicules. En effet, on peut ne pas avoir le temps de planifier des aller-retours. Comme la flotte de véhicule est limitée pour chaque type de véhicule, on peut alors être obligé d'utiliser plusieurs types de véhicules pour servir

la demande. Considérons un exemple : une demande n_1 de $q_{n_1} = 75t$ a une contrainte horaire largement inférieure à l'horizon de temps sur sa collecte, c'est à dire que $l_{p_{n_1}} - e_{p_{n_1}} \lll \bar{H}$. La flotte de véhicules possède uniquement deux véhicules $n_{f_1} = 2$ d'une même catégorie f_1 de capacité unitaire de $Q^{f_1} = 25t$. L'horizon de temps corrigé est de 8 heures $\bar{H} = 8$. La durée d'une rotation de type Ω_1 faite par un véhicule f_1 pour satisfaire la demande n_1 est $T_{f_1 n_1} = 3h$.

Le modèle $\mathcal{P}_1(\Omega_1)$ retourne une solution optimale avec $X_{f_1 n_1} = 3$ soit trois rotations à effectuer par les deux véhicules de la catégorie f_1. Cet exemple est illustré dans la figure 4.



Exemple 4 – Exemple de l'utilisation des bornes. Le véhicule en trait plein ne peut pas réaliser une deuxième rotation dans la fenêtre de temps impartie. Il est nécessaire d'utiliser un deuxième véhicule (trait pointillé) pour pouvoir effectuer une livraison.

Nous remarquons que le premier véhicule en trait plein ne pourra faire une autre collecte dans la fenêtre de temps. Sur cet exemple, le nombre de requêtes que nous pouvons effectuer avec ces deux véhicules est de 2, ce qui conduit à une absence de solution lors de la réalisation des tournées par la seconde phase $\mathcal{P}_2$.

Pour limiter ce type de problématique, nous introduisons des bornes supérieures sur les variables X_{fn}, indiquant le nombre maximum de rotations pouvant être réalisées sur la fenêtre de temps de la tournée avec un véhicule de la catégorie f.

Pour calculer cette borne, nous devons calculer le maximum de livraisons réalisables pour une demande n par un véhicule de la catégorie f par rapport à la fenêtre de temps au point de collecte $TW_n^p = l_{p_n} - e_{p_n}$ et la fenêtre de temps au point de livraison $TW_n^d = l_{d_n} - e_{d_n}$.

Il faut tenir compte d'une période ajustée pour la collecte (respectivement la livraison) en ajoutant le temps de service à la livraison (resp. à la collecte) et les temps de trajets car ceux-ci peuvent s'effectuer en dehors de la fenêtre de temps de la collecte (resp. de la livraison). Nous nommons cette période ajustée T^p_{global} pour la collecte et T^d_{global} pour la livraison. La durée nécessaire d'une rotation est T_ω.

$$T^p_{global} = TW^p_n + t^f_{p_n d_n} + s_{d_n} + t^f_{d_n p_n}$$

$$T^d_{global} = TW^d_n + s_{p_n} + t^f_{p_n d_n} + t^f_{d_n p_n}$$

Pour une demande n et une catégorie de véhicules f, le nombre maximum de rotations $maxRot_{fn}$ que peut effectuer un véhicule est donné par :

$$maxRot_{fn} = \lfloor \frac{\min(T^p_{global}, T^d_{global})}{T_\omega} \rfloor$$

La figure 5.4 représente cette borne pour le cas de la collecte et la figure 5.5 pour le cas de la livraison. Les tournées foncées représentent l'ensemble des tournées pouvant être effectuées dans la fenêtre de temps.

Ainsi, la borne maximum sur les rotations que peuvent faire les véhicules de la catégorie f pour servir la demande n est défini par les contraintes (5.5).

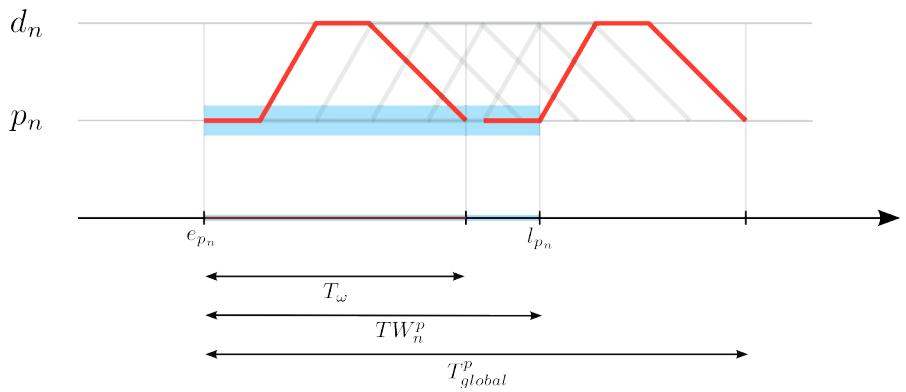


FIGURE 5.4 – Représentation du nombre d'aller-retour maximum dans le cadre d'une collecte. Dans la fenêtre de temps TW_n^p, 6 véhicules différents peuvent se succéder. Également, le premier véhicule peut effectuer la première rotation et la dernière. Son nombre maximal de rotations est de 2.

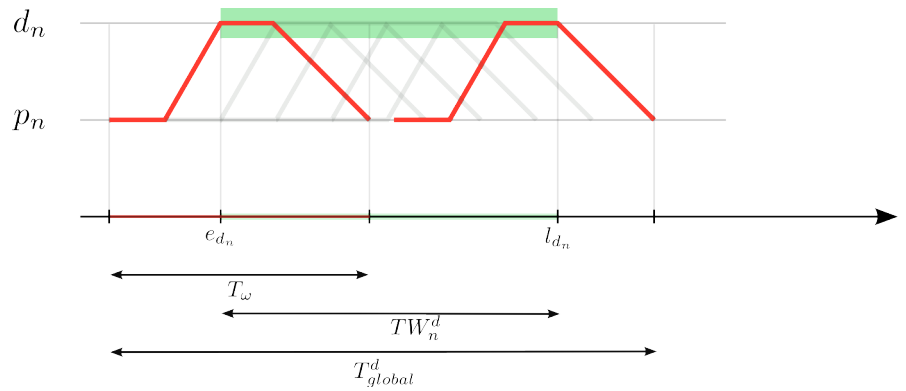


FIGURE 5.5 – Représentation du nombre d'aller-retour maximum dans le cadre d'une livraison. Dans la fenêtre de temps TW_n^p, 6 véhicules différents peuvent se succéder. Également, le premier véhicule peut effectuer la première rotation et la dernière. Son nombre maximal de rotations est de 2.

$$X_{fn} \leq nbVehicle_f * maxRot_{fn} \qquad \forall n \in N, \forall f \in F \tag{5.5}$$

Cette contrainte est ajoutée au modèle $\mathcal{P}_1(\Omega_1)$.

5.2.3 Modèle monopériodique avec rotations composées

Dans ce modèle, nous considérons les rotations de type Ω_1 et Ω_2. Les notations utilisées sont identiques au modèle précédent. Une rotation $\omega_1 \in \Omega_1$ lie une demande n à une catégorie de véhicules f. Une rotation $\omega_2 \in \Omega_2$ lie les demandes n_1 et n_2 à une catégorie de véhicules f.

Nous introduisons les variables entières $X_\omega \in \mathbb{N}$ représentant le nombre de fois que la rotation $\omega \in \Omega$ est effectuée.

La modélisation mathématique du problème $\mathcal{P}_1(\Omega_{1,2})$ est la suivante :

$$\mathcal{P}_1(\Omega_{1,2})$$

$$\min \quad \sum_{\omega \in \Omega} C_\omega X_\omega \tag{5.6}$$

$$s.c.$$

$$\sum_{\omega \in \Omega^{f//}} T_\omega * X_\omega \leq \bar{H} * nbVehicle_f \qquad \forall f \in F \tag{5.7}$$

$$\sum_{\omega \in \Omega^{/n/}} Q_\omega * X_\omega \geq q_n \qquad \forall n \in N \tag{5.8}$$

$$X_\omega \in \mathbb{N}^+ \tag{5.9}$$

La fonction objectif du problème (5.6) est de minimiser le coût des rotations sélectionnées entre les véhicules et les demandes. Les contraintes (5.7) limitent le temps total d'utilisation des véhicules pour chaque catégorie de véhicules. Les contraintes (5.8) garantissent que la quantité de produits livrée pour une demande par toutes les catégories de véhicules est respectée.

Cette formulation est similaire avec le modèle précédent dans sa structure : contraintes temporelles et contraintes de capacité. Pour autant, il est plus fort dans la qualité de la solution car il permet de regrouper les demandes qui présentent un potentiel d'optimisation entre elles.

5.3 Modèle multipériodique

Une extension du modèle $\mathcal{P}_1(\Omega_{1,2})$ est la version multipériodique. Nous considérons un horizon multipériodique noté T, constitué de périodes t de durée H_t. La différence réside dans le fait qu'une demande peut être servie sur plusieurs périodes. Nous supposons que la flotte de véhicules est identique pour chaque période. L'ensemble T_n indique les périodes pour lesquelles la demande n peut être servie. L'enjeu est de prévoir ensemble les transports qui apportent un potentiel d'optimisation.

Chaque rotation $\omega \in \Omega$ livre une quantité Q_ω de produits et consomme une durée T_ω. L'ensemble des rotations pouvant servir la demande $n \in N$ à la période $t \in T$ est noté $\Omega^{/n/t} \subset \Omega$. Par analogie, l'ensemble des rotations pouvant être parcourues par un véhicule de la catégorie $f \in F$ à la période $t \in T$ est noté $\Omega^{f//t} \subset \Omega$. Le coût d'une rotation $\omega \in \Omega$ est noté C_ω.

Pour modéliser ce problème, une variable de décision positive et entière X^t_ω définit le nombre total de fois que la rotation ω est utilisée lors de la période t.

La modélisation mathématique du problème $\mathcal{P}_1(\Omega_{1,2}$ / $T)$ est la suivante.

$\mathcal{P}_1(\Omega_{1,2}\,/\,T)$

$$\min \quad \sum_{\omega\in\Omega}\sum_{t\in T} C_\omega X^t_\omega \tag{5.10}$$

$s.c.$

$$\sum_{\omega\in\Omega^{f//t}} T_\omega * X^t_\omega \leq H^t * nbVehicle^t_f \qquad \forall f \in F\ \forall t \in T \tag{5.11}$$

$$\sum_{t\in T_n}\sum_{\omega\in\Omega^{/n/t}} Q_\omega * X^t_\omega \geq q_n \qquad \forall n \in N \tag{5.12}$$

$$X^t_\omega \in \mathbb{N}^+ \tag{5.13}$$

L'objectif du problème (5.10) est de minimiser le coût des rotations d'affectations des véhicules aux demandes. Les contraintes (5.11) limitent le temps total d'utilisation des véhicules pour chaque catégorie de véhicules en prenant en compte le temps d'une rotation de type Ω_1 et Ω_2. La quantité de produits livrée pour une demande par toutes les rotations valides doit être respectée, contraintes (5.12).

Ainsi le modèle proposé pour séparer les demandes en requêtes permet donc également de séparer les demandes sur les différentes périodes ou journées de l'horizon temporel. L'introduction de rotation à deux requêtes permet ici de modéliser que deux demandes qui peuvent être combinées avantageusement doivent être placées sur la même période. La rotation à deux requêtes est moins coûteuse que les sommes des rotations à une requête et consomme également moins de ressources temporelles.

5.4 Conclusion

Nous avons proposé une nouvelle approche de découpage des demandes en requêtes. Cette approche décide de l'affectation des requêtes aux catégories de véhicules en amont de la phase $\mathcal{P}_2$ de création de tournées. Ceci constitue une différence avec les travaux de la littérature [96, 95, 8].

D'une part, la modélisation, similaire au TP, permet une résolution aisée par les principaux solveurs. Le découpage repose sur une approche heuristique du temps consommé. Nous prévoyons ainsi que la phase $\mathcal{P}_2$ ne puisse pas insérer toutes les requêtes. Une étape importante est la création des requêtes une fois la phase $\mathcal{P}_1$ résolue. Cette étape est abordée ultérieurement dans le chapitre 6.

Une perspective à cette méthode est de pouvoir revenir sur le découpage après analyse des requêtes non insérées et des véhicules non utilisés.

6

Méthodologie de création des requêtes

Sommaire

Dans ce chapitre, nous nous intéressons à l'algorithme de création des requêtes. Cette opération n'est pas triviale mais est une combinaison du résultat de la première phase et des règles dites *métiers* de l'entreprise. Dans un premier temps, dans la section 6.1, nous positionnons cette composante dans notre algorithme global. Nous détaillons la création des requêtes dans la section 6.2. Enfin, nous concluons dans la section 6.3.

6.1 Intégration de la méthodologie de création des requêtes dans notre algorithme

La première phase, présentée dans le chapitre 5, fournit un ensemble de variables X_{fn} ou X_{ω} représentant la quantité de trajets à effectuer par les véhicules pour une demande d'une période donnée. Lorsqu'il s'agit d'une variable X_{fn}, elle exprime le nombre de trajets que doit effectuer un véhicule de la catégorie $f \in F$ pour satisfaire la demande $n \in N$. Lorsqu'il s'agit d'une variable X_{ω}, cette variable exprime le nombre de trajets d'une rotation $\omega \in \Omega$, qui est l'expression d'une ou plusieurs demandes servies par un véhicule.

Chaque rotation est dès lors traduite en une requête. La création des requêtes à partir des variables exprimant les rotations est effectuée par la méthode $\mathrm{createRequest}$ qui est une composante de l'algorithme. La méthode $\mathrm{createRequest}$ utilise le résultat de la phase $\mathcal{P}_1$ en données d'entrée. En fonction de ces données, elle crée un ensemble R de requêtes devant être servies par les véhicules. Ces requêtes constituent la sortie de cette méthode et servent comme données d'entrée pour la résolution de la phase $\mathcal{P}_2$.

6.2 Description de la création des requêtes

La méthode $\mathrm{createRequest}$ crée X_{fn} requêtes à partir de la demande $n \in N$ et pour la catégorie de véhicules $f \in F$. Notre problème industriel comporte deux types de demande : (i) les demandes à flux détendus $N_{\mathcal{U}} \subset N$ et (ii) les demandes à flux tendus $N_{\mathcal{S}} \subset N$. En fonction des caractéristiques de ces demandes, le découpage en requête s'effectue de façons différentes.

Nous décrivons dans un premier temps les caractéristiques d'une requête dans la section 6.2.1. Puis, nous détaillons les différents types des demandes considérées dans la section 6.2.2. Enfin, nous présentons la création des requêtes pour les demandes à flux détendus, section 6.2.3, puis la création des requêtes pour les demandes à flux tendus, section 6.2.4.

6.2.1 Caractéristique d'une requête

Une requête est le trajet d'un camion servant une partie, ou la totalité, d'une demande en effectuant deux opérations dans l'ordre suivant : (i) une collecte et (ii) une livraison. Chaque requête est définie par une fenêtre de temps pour la collecte et une fenêtre de temps pour la livraison. La quantité de matériaux transportée est exprimée en tonnes (t).

Les opérations de collecte et de livraison s'effectuent sur des ressources. Pour une opération, chaque camion utilise, pour une certaine durée, la ressource sur laquelle il se trouve en plus du temps passé dans sa tournée. Nous distinguons deux durées de service pour chaque opération d'une requête et dépendantes du type de demande dont est issue la requête :

- la durée de service sur la tournée s^{route} : c'est la durée réelle totale que prend l'opération à s'effectuer. Elle est équivalente au temps que passe le camion sur le site (durée de chargement/déchargement, durée de pesée, temps d'attente, etc.).
- la durée de service sur la ressource $s^{resource}$: c'est la durée d'occupation de la ressource inhérente au service de la requête. En fonction du type de ressource, le camion occupe plus ou moins de temps sur la ressource.

La durée de service sur la ressource est toujours inférieure ou égale à la durée de service sur la tournée.

$$s^{resource} \leq s^{route} \tag{6.1}$$

Nous détaillons dans le tableau 6.1 les différentes caractéristiques des attributs d'une requête r.

Le problème industriel que nous résolvons nous a amenés à faire des choix de modélisation pour définir l'occupation d'une requête sur une ressource. Sur les ressources de l'entreprise (plateformes, sites, carrières, etc.), l'opération critique est la pesée des camions sur les ponts-bascules qui doit être effectuée avant et après le chargement/déchargement du camion. Nous définissons une durée constante de pesée sur un pont-bascule définie par le paramètre suivant : $s^{weighbridge}_{default}$. Les deux modélisations possibles pour la durée passée par une requête sur une ressource sont les suivantes :

1. Occupation uniquement du pont-bascule Lorsqu'un camion effectue une opération de chargement ou de déchargement sur une ressource, le camion effectuera un passage sur le pont-bascule en arrivant sur le site et en repartant du site. Ces opérations de pesées obligatoires constituent une utilisation de la ressource. C'est l'utilisation courante de la ressource. La durée de pesée par défaut passée sur un pont-bascule est $s^{weighbridge}_{default}$ et est égale à deux fois la durée d'une pesée sur un pont-bascule. Cette hypothèse est réaliste puisque nous considérons que l'ordre des camions ne change pas sur une ressource.

2. Occupation continue d'une ressource Sur des ressources où l'occupation est continue (demande à flux tendu), la durée de service sur la ressource est égale au temps total d'utilisation de la ressource par le camion.

Attribut	Valeur	Unités
p_r	point de livraison	-
d_r	point de collecte	-
e_{p_r}	heure d'ouverture du service à p_r	min
l_{p_r}	heure de fermeture du service à p_r	min
e_{d_r}	heure d'ouverture du service à d_r	min
l_{d_r}	heure de fermeture du service à d_r	min
$s_{p_r}^{route}$	durée de service sur la tournée de la collecte	min
$s_{d_r}^{route}$	durée de service sur la tournée de la livraison	min
$s_{p_r}^{resource}$	durée de service sur la ressource de la collecte	min
$s_{d_r}^{resource}$	durée de service sur la ressource de la livraison	min
π_{p_r}	ressource de la collecte	-
π_{d_r}	ressource de la livraison	-
q_r	quantité de matière à transporter	t

Tableau 6.1 – Description des attributs d'une requête.

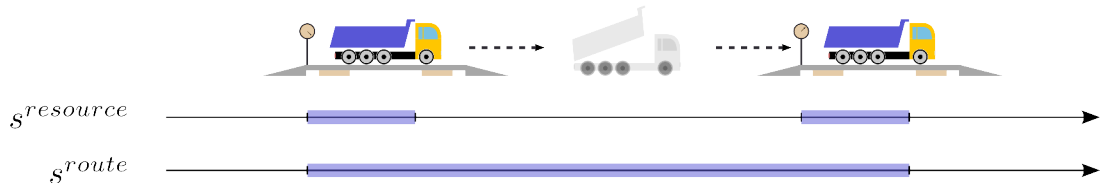


FIGURE 6.1 – Description des durées de service sur la ressource et la tournée pour une occupation uniquement du pont-bascule.

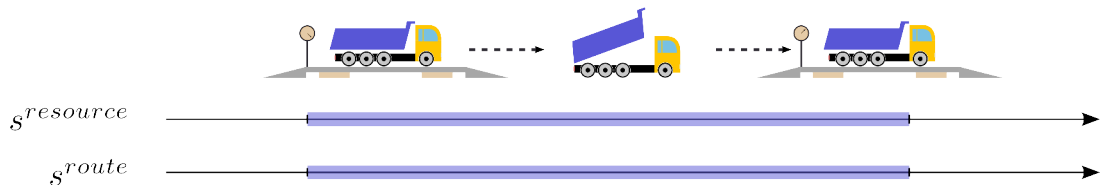


FIGURE 6.2 – Description des durées de service sur la ressource et la tournée pour une occupation continue de la ressource.

6.2.2 Caractéristique des demandes de notre problème

Les demandes de l'entreprise sont diverses et variées. Nous proposons plusieurs attributs pour prendre en compte cette diversité et permettre de couvrir tous les besoins de l'entreprise. Chaque requête est définie en fonction des attributs de la demande à laquelle elle est associée. Nous devons définir, premièrement, les attributs de la requête (durée de service, ressource) et deuxièmement, les fenêtres de temps de la requête.

Outre les attributs déjà définis d'une demande dans la section 5.1, des attributs complémentaires permettent de prendre en compte la diversité. Ils sont listés dans le tableau 6.2.

Attribut	Type	Unité
$pickupResource_n$	$boolean$	-
$deliveryResource_n$	$boolean$	-
$pickupDurationService_n$	$double$	min
$deliveryDurationService_n$	$double$	min
$pickupLean_n$	$boolean$	-
$deliveryLean_n$	$boolean$	-
$pickupLeanCapacity_n$	$double$	t/h
$deliveryLeanCapacity_n$	$double$	t/h
$directionLean_n$	$string$	-

Tableau 6.2 – Attributs complémentaires d'une demande n. Dans la colonne Type, $boolean$ indique une donnée de type booléen qui prend la valeur $vrai/faux$. $double$ indique un type nombre réel. $string$ indique une chaîne de caractère.

Ces attributs permettent de définir la valeur des durées de service et la ressource sur laquelle a lieu la requête. Leurs définitions sont les suivantes :

- $pickupResource_n$ (respectivement $deliveryResource_n$) : indique si la demande n est associée à une ressource ($true$) ou non ($false$) au point de collecte (resp. de livraison) ;
- $pickupDurationService_n$ (respectivement $deliveryDurationService_n$) : indique la durée nécessaire pour effectuer une opération sur le point de collecte (resp. de livraison) ;
- $pickupLean_n$ (respectivement $deliveryLean_n$) : indique si les requêtes de la demande n occupent le pont-bascule ($false$) ou occupent la ressource de façon continue ($true$) au point de collecte (resp. de livraison) ;
- $pickupLeanCapacity_n$ (respectivement $deliveryLeanCapacity_n$) : indique le cadencement de l'opération de la demande n, en t/h, sur la ressource de la collecte (resp. de la livraison) ;
- $directionLean_n$: indique la direction de la demande n : "I" pour un apport de matériaux sur le chantier, "O" pour un enlèvement de matériaux sur le chantier.

Les demandes de notre problème industriel appartiennent à l'ensemble N. Les demandes de N sont divisées en deux sous-ensembles : (1) $N_{\mathcal{U}}$ pour les demandes à flux détendus et (2) $N_{\mathcal{S}}$ pour les demandes à flux tendus.

Lors de leurs créations, les requêtes héritent des fenêtres de temps de la demande à laquelle elles sont associées. C'est le cas général de fonctionnement et il correspond aux demandes à flux détendus $N_{\mathcal{U}}$.

En revanche, pour les demandes à flux tendus, comme les enrobés, il est nécessaire de définir de nouvelles fenêtres de temps pour la collecte et la livraison afin de créer un service sans discontinuité. Cela correspond aux demandes $N_{\mathcal{S}}$.

Dans la suite de ce document, nous détaillons le calcul de ces fenêtres de temps.

6.2.3 Création des requêtes pour les demandes à flux détendus $n \in N_{\mathcal{U}}$

Les requêtes à flux détendus sont essentiellement des transports en vrac. Pour ces requêtes, la collecte et la livraison peuvent être faites à n'importe quels moments dans les fenêtres de temps de la demande associée. Nous considérons une requête r créée à partir d'une demande $n \in N_{\mathcal{U}}$ et pour une catégorie de véhicules f.

Pour les requêtes à flux détendus, les attributs d'une demande n ont les valeurs suivantes : $pickupLean_n = false$ et $deliveryLean_n = false$. Le tableau 6.3 présente le choix effectué pour les attributs.

Attribut	Valeur
p_r	p_n
d_r	d_n
e_{p_r}	e_{p_n}
l_{p_r}	l_{p_n}
e_{d_r}	e_{d_n}
l_{d_r}	l_{d_n}
$s^{route}_{p_r}$	$pickupDurationService_n$
$s^{route}_{d_r}$	$deliveryDurationService_n$
$s^{resource}_{p_r}$	$2 * s^{weighbridge}_{default}$
$s^{resource}_{d_r}$	$2 * s^{weighbridge}_{default}$
q_r	Q_f

Tableau 6.3 – Valeurs des attributs d'une requête dans le cadre d'une demande à flux détendu. Nous considérons deux passages sur le pont-bascule : un en entrée et un en sortie.

Même si la catégorie des véhicules impliquent des capacités différentes de camions, il a été remarqué, par l'entreprise que le temps de chargement ou de déchargement d'un camion servant une demande $n \in N_{\mathcal{U}}$ ne variait pas beaucoup suivant le type de camion. C'est la typologie de la ressource qui définissait ce temps moyen. C'est pour cette raison que le temps de service sur la tournée est égal au temps de service moyen de la demande considérée.

6.2.4 Création des requêtes pour les demandes à flux tendus $n \in N_{\mathcal{S}}$

Les requêtes à flux tendus, d'une demande $n \in N_{\mathcal{S}}$, sont caractérisées par une cadence de service, exprimée en t/h, au point de collecte $pickupLeanCapacity_n$ et au point de livraison $deliveryLeanCapacity_n$. Ces requêtes occupent de façon continue les ressources concernées (attribut $pickupLean_n$ à ***true***).

La borne inférieure d'une opération, pour la collecte e_{p_n} et pour la livraison e_{d_n}, indique l'heure à laquelle le service de l'opération commence sur la ressource. À partir de cette borne, les camions se succèdent pour servir cette demande. La durée totale de service de l'opération est alors inférieure ou égale à la largeur initiale de la fenêtre de temps de la demande. Elle dépend du nombre de camions affectés au service de cette demande.

Si les requêtes associées à une demande n à flux tendu sur la collecte et/ou la livraison conservent la même fenêtre de temps, il peut y avoir des discontinuités dans le service, ce qui serait contraire à la définition de cette demande. Cet exemple est illustré par la figure 6.3.

La durée d'occupation d'un camion de la ressource est directement liée à la capacité du camion. Un camion de grande capacité occupera plus longtemps la ressource qu'un camion de petite capacité lorsque nous considérons un temps de cadencement identique, ce qui est le cas dans notre problème. Ainsi, nous ne pouvons pas ordonner les requêtes sur la ressource puisque leurs durées dépendent des camions.

Pour cela, nous choisissons de réduire la fenêtre de temps des requêtes pour permettre un service sans discontinuité. Nous présentons la méthode retenue pour calculer la fenêtre de temps pour un service de type collecte (la méthode est identique pour la livraison).

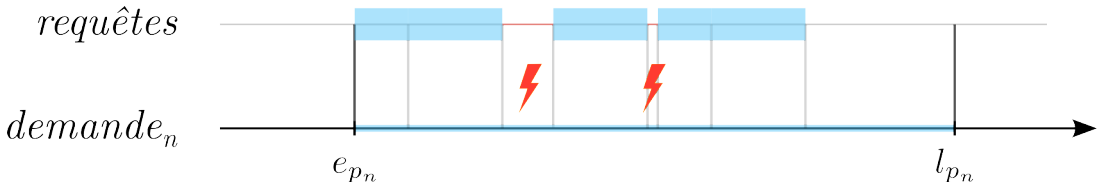


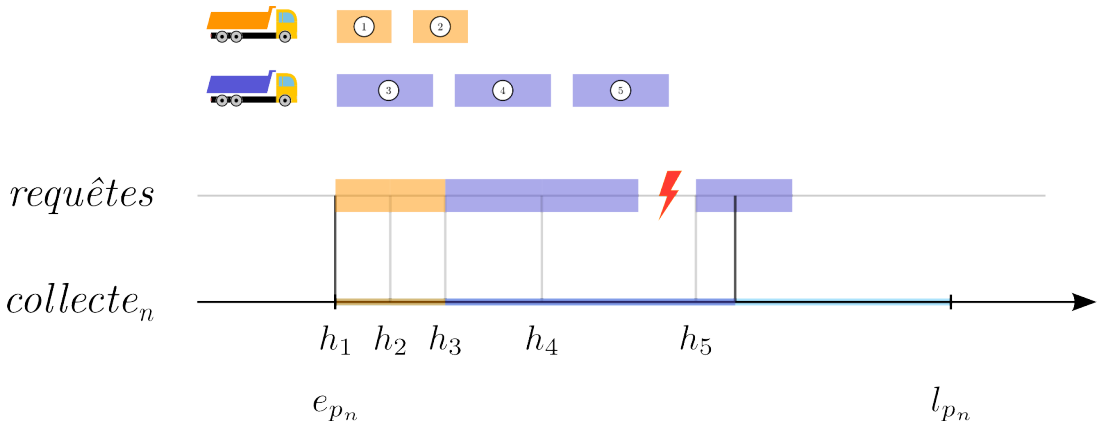
FIGURE 6.3 – Discontinuité dans le service d'une demande à flux tendu. Les espaces entre les requêtes (symbolisés par les éclairs) représentent une discontinuité.

Chaque variable X_{fn} de la première phase représente le nombre de requêtes à créer pour la demande n et la catégorie de véhicules f. La durée de service, en minutes, d'un véhicule de la catégorie f pour la collecte de la demande n, représentée par la variable s^{nf}_{pickup}, vaut $s^{nf}_{pickup} = 60 * Q^f / pickupLeanCapacity_n$. La durée totale de service pour la collecte s^n_{pickup} est donnée par l'équation 6.2.

$$s^n_{collecte} = \sum_{f \in F} s^{nf}_{pickup} * X_{fn} \tag{6.2}$$

Ainsi, l'ensemble des requêtes associées à la collecte devra avoir son heure de début de service dans l'intervalle $[e_{p_n}, e_{p_n} + s^n_{pickup}]$ pour chacune d'entre elle. Malgré cela, cet intervalle reste trop large et permet à la dernière requête, par exemple, de commencer en décalage et créer de la discontinuité.

Nous prenons un exemple pour illustrer cette méthode où la collecte d'une demande n doit être servie par 2 requêtes (1 et 2) de véhicule de type orange et 3 requêtes (3, 4 et 5) de véhicules de type violet. La figure 5 présente le cas où la dernière requête peut créer une discontinuité dans le service.



Exemple 5 – Exemple de discontinuité créée par le service de la requête 5.

Pour cela, nous décidons de réduire l'intervalle du minimum des temps de service de la collecte de la demande n pour chaque catégorie de véhicules f. Ainsi la fenêtre de temps devient :

$$[e_{p_n}, e_{p_n} + s^n_{pickup} - \min_{f \in F} s^{nf}_{pickup}]$$

La figure 6.4 montre la solution retenue.

Dans une solution, où toutes les requêtes doivent être satisfaites, cette méthode imposera que la dernière requête servie pour une demande soit une requête avec le plus petit temps de service. Cette méthode est cohérente avec les pratiques de l'entreprise où le service se fait, dans un premier temps, avec les camions de plus forte capacité, donc de plus grand temps de service.

Le tableau 6.4 présente les valeurs des attributs d'une requête r d'une demande à flux tendu.

Le temps de service sur la tournée est égal au temps de service sur la ressource ce qui correspond à une utilisation continue de la ressource.

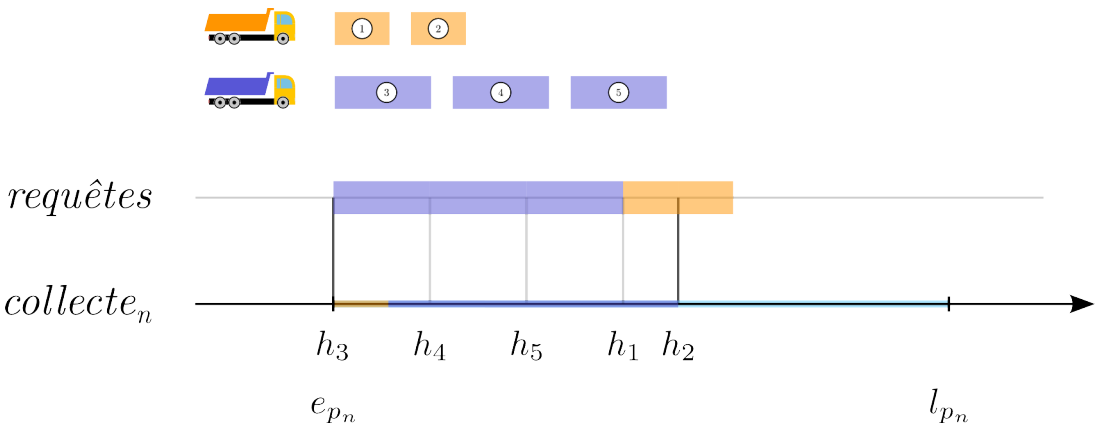


FIGURE 6.4 – Solution retenue pour la création des requêtes.

Attribut	Valeur
p_r	p_n
d_r	d_n
e_{p_r}	e_{p_n}
l_{p_r}	l_{p_n}
e_{d_r}	e_{d_n}
l_{d_r}	l_{d_n}
$s_{p_r}^{route}$	$60 * Q_f / pickupLeanCapacity_n$
$s_{d_r}^{route}$	$60 * Q_f / deliveryLeanCapacity_n$
$s_{p_r}^{resource}$	$60 * Q_f / pickupLeanCapacity_n$
$s_{d_r}^{resource}$	$60 * Q_f / deliveryLeanCapacity_n$
q_r	Q_f

Tableau 6.4 – Valeurs des attributs d'une requête dans le cadre d'une demande à flux tendu.

6.2.5 Cas des demandes sans ressource sur la collecte et/ou la livraison

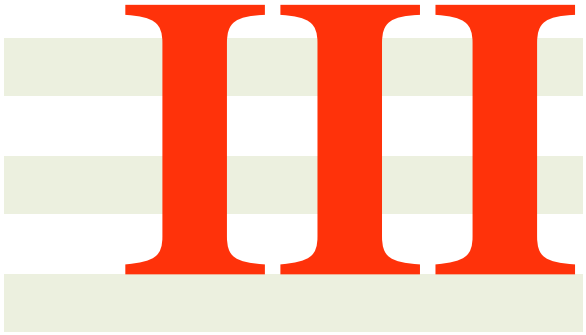
Pour le cas d'une demande $n \in N$ qui n'est associée à aucune ressource (l'attribut $pickupResource_n$ est à $false$), l'opération de collecte n'est attribuée à aucune ressource. Le début de service de l'opération de collecte doit être effectué dans la fenêtre de temps mais il n'y a pas de temps de service défini sur la ressource. Les opérations de collecte de ces demandes n'ont aucune précédence entre elles. En revanche, le temps de service sur la tournée d'une requête est égale au temps de service de la demande $pickupDurationService_n$.

Le même raisonnement est fait pour le point de livraison.

6.3 Conclusion

Ce chapitre a présenté la méthodologie que nous utilisons pour créer l'ensemble R de requêtes à partir des résultats fournis par la partie $\mathcal{P}_1$. Les paramètres définissant les demandes permettent de créer une grande variété de requêtes en caractérisant leurs fenêtres de temps mais aussi leurs temps de service. Pour notre problème, nous avons créé des requêtes à flux tendus (provenant de l'ensemble de demandes $N_{\mathcal{S}}$) et des requêtes à flux détendus (provenant de l'ensemble de demandes $N_{\mathcal{U}}$). Notre modélisation permet la création de requêtes sans ressource. Ce dernier cas permet de traiter le problème de FTPDPTW.

La considération de cette variété permet de satisfaire les besoins exprimés par l'entreprise dans le cadre de notre projet.



Recherche de solutions pour le Rich FT-PDP-RS

Table des matières

Introduction et présentation du modèle Rich-FT-PDP-RS

Dans la partie précédente, nous nous sommes intéressés au découpage des demandes, ce qui correspond à la première phase $\mathcal{P}_1$ de la méthode de résolution. Le chapitre 6 a présenté la création des requêtes et de leurs fenêtres de temps. Connaissant les requêtes et leurs fenêtres de temps, nous devons, à présent, résoudre un problème de tournées de véhicules.

Dans cette partie, nous présentons le modèle Rich-FT-PDP-RS. Ce modèle englobe toutes les contraintes du problème. C'est une modélisation complète du problème industriel incluant les différentes thématiques de ce travail de recherche, à savoir :

- tournées de collectes et livraisons ;
- synchronisation aux ressources ;
- temps de conduite des chauffeurs ;
- pauses-déjeuner.

Le problème de tournées de véhicules que nous définissons dans cette partie est un problème où un ensemble de requêtes doit être servi par des véhicules. Chaque véhicule, appartenant à une catégorie donnée, réalise une tournée en servant des requêtes composées séquentiellement d'une collecte et d'une livraison. Suite au découpage de la phase précédente, chaque requête est considérée comme un transport en camion plein.

Les opérations des requêtes (collecte et livraison) sont réalisées sur des sites industriels de l'entreprise. Chaque opération s'effectue sur une ressource rattachée à un site. Sur chaque ressource et à chaque instant, une et une seule opération peut être servie.

Les chauffeurs des véhicules doivent respecter un temps maximum de conduite par période. De plus, chaque chauffeur ne peut pas excéder un temps maximum de conduite sans faire de pause. Dans notre cas, considérant une amplitude de travail équivalente à une journée, cette pause est la pause-déjeuner. Ainsi, chaque chauffeur est limité dans la conduite par un temps maximum avant la pause-déjeuner, par un temps maximum après la pause-déjeuner et par un temps maximum sur toute la période. La pause-déjeuner a lieu sur le site de collecte ou de livraison d'une requête.

Nous nommons $\mathcal{P}_2$ le problème de tournées de véhicules associé au service des requêtes et prenant en compte toutes les contraintes précédemment citées. Nous modélisons ce problème avec les notations suivantes.

Notations Nous considérons un ensemble R de n requêtes de transport devant être servies par une flotte hétérogène K comportant $|K|$ véhicules hétérogènes. Chaque requête $r \in R$ est représentée par un point de collecte $p_r \in P$ et un point de livraison $d_r \in D$. Le début d'un point $i \in P \cup D$ de collecte ou de livraison doit se faire dans la fenêtre de temps $[e_i, l_i]$. La durée du service à un sommet $i \in P \cup D$, point de collecte ou de livraison, est notée s_i. L'ensemble des véhicules pouvant servir une requête $r \in R$ est noté $K^r \subset K$.

Un véhicule $k \in K$ commence sa tournée à un dépôt d'origine $o_1(k) \in O_1$ et termine sa tournée à un dépôt d'arrivée $o_2(k) \in O_2$. L'ensemble des dépôts est noté $O = O_1 \cup O_2$. Chaque dépôt $o \in O$ est ouvert pendant une fenêtre de temps $[e_o, l_o]$.

Chaque chauffeur d'un véhicule doit effectuer une pause-déjeuner lors de sa tournée. Un chauffeur étant affecté à un seul véhicule durant sa tournée, nous confondons volontairement les chauffeurs et les véhicules lorsque nous évoquons la pause-déjeuner. L'ensemble des pauses-déjeuner est noté L. Le début d'une pause-déjeuner $l \in L$ doit être effectué dans une fenêtre de temps $[e_l, l_l]$ et la durée de la pause-déjeuner est s_{lunch}. La tournée d'un véhicule ne doit pas dépasser un temps maximum T^{daily}. Le temps de trajet consécutif d'un véhicule avant et après une pause-déjeuner ne doit pas dépasser $T^{interval}$.

Le problème $\mathcal{P}_2$ est défini sur un graphe orienté $G = (V, A)$. L'ensemble des sommets V regroupe les dépôts, les points de collecte, les points de livraison et les points de déjeuner $V = O \cup P \cup D \cup L$. L'ensemble des arcs pouvant être parcourus par les véhicules est le suivant : $A = \{(o_1(k), o_2(k))|k \in K\} \cup O_1 \times P \cup \{(p_r, d_r)|r \in R\} \cup D \times O_2 \cup D \times P \cup D \times L \cup L \times P$.

L'ensemble des arcs du problème est synthétisé dans la figure 6.5.

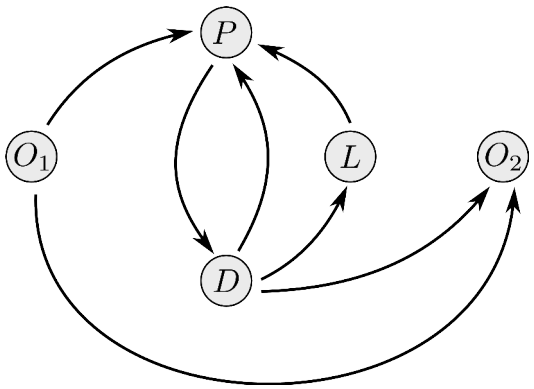


FIGURE 6.5 – Graphe des arcs du problème $\mathcal{P}_2$.

Les temps de trajets entre les points dépendent des catégories de véhicules et des caractéristiques de la route. La matrice de distance est calculée depuis des données réelles issues d'un SIG, ce qui génère des distances asymétriques entre deux points. Le temps de trajet entre deux points $(i, j) \in A$ parcouru par un véhicule $k \in K$ est noté t_{ij}^k et la distance est notée d_{ij}^k. Particulièrement, le temps de trajet et la distance du trajet d'un véhicule $k \in K$ entre son dépôt d'origine et son dépôt d'arrivée sont nuls : $t_{o(k)o'(k)} = d_{o(k)o'(k)} = 0$.

Le transport des produits engendre plusieurs coûts. Tout d'abord, le coût fixe d'utilisation d'un véhicule $k \in K$ est noté $\mathcal{CD}^k$. Le coût kilométrique lié aux transports est noté $\mathcal{CK}^k$, coût par km parcouru par type de véhicule et le coût horaire lié au temps de travail est noté $\mathcal{CH}^k$.

L'ensemble des ressources liées aux requêtes est noté Π et l'ensemble des opérations de collecte ou de livraison devant s'effectuer sur une ressource $\pi \in \Pi$ est noté $V_\pi \subset P \cup D$. Sur chaque ressource, une et une seule opération peut être réalisée à un instant donné.

Variables Nous définissons la variable y^k qui est égale à 1 si le véhicule $k \in K$ effectue une tournée dans la solution, 0 sinon. La variable x_{ij}^k est égale à 1 si un véhicule $k \in K$ traverse l'arc $(i, j) \in A$, 0 sinon. L'heure de début de service à un sommet $i \in V$ est modélisé par la variable entière h_i. La variable z_{ij}^π est égale à 1 si le sommet $i \in V_\pi$ est planifié avant le point $j \in V_\pi$ sur la ressource π. Une variable $drive_i^{k,before}$ représente le temps cumulé de trajet depuis le début de la période jusqu'à la pause-déjeuner pour chaque point $i \in V$ pour le véhicule $k \in K$. La variable $drive_i^{k,after}$ représente le temps de trajet cumulé depuis la fin de la pause-déjeuner jusqu'à la fin de période pour chaque point $i \in V$ pour le véhicule $k \in K$.

Modèle Le modèle mathématique du problème $\mathcal{P}_2$ est décrit de la manière suivante :

$$\min \sum_{k \in K} \mathcal{CD}^k * y^k + \sum_{k \in K} \sum_{(i,j) \in A} \mathcal{CK}^k * x_{ij}^k * d_{ij}^k + \sum_{k \in K} \mathcal{CH}^k * (h_{o_2(k)} - h_{o_1(k)}) \tag{6.3}$$

$s.t.$

$$\sum_{(o_1(k),i) \in A} x_{o_1(k)i}^k = \sum_{(i,(o_2(k)) \in A} x_{io_2(k)}^k \qquad \forall k \in K \tag{6.4}$$

$$\sum_{(o_1(k),i) \in A} x_{o_1(k)i}^k = y^k \qquad \forall k \in K, i \neq o_2(k) \tag{6.5}$$

$$\sum_{(j,i) \in A} x_{ji}^k = \sum_{(i,j) \in A} x_{ij}^k \qquad \forall k \in K, \forall i \in V \tag{6.6}$$

$$\sum_{k \in K_r} x_{p_r d_r}^k = 1 \qquad \forall r \in R \tag{6.7}$$

$$h_j \geq (h_i + s_i + t_{ij}^k) x_{ij}^k \qquad \forall (i,j) \in A, \forall k \in K \tag{6.8}$$

$$e_i \leq h_i \leq l_i \qquad \forall i \in V \tag{6.9}$$

$$h_j \geq (h_i + s_i) z_{ij}^\pi \qquad \forall \pi \in \Pi, \forall i \in V_\pi, \forall j \in V_\pi \tag{6.10}$$

$$z_{ij}^\pi + z_{ji}^\pi = 1 \qquad \forall \pi \in \Pi, \forall i \in V_\pi, \forall j \in V_\pi \tag{6.11}$$

$$h_{o_2(k)} - h_{o_1(k)} - s_{lunch} \leq T^{daily} \qquad \forall k \in K \tag{6.12}$$

$$drive_{o_1(k)}^{k,before} = 0 \qquad \forall k \in K \tag{6.13}$$

$$x_{ij}^k = 1 \Rightarrow drive_j^{k,before} \geq t_{ij}^k + drive_i^{k,before} \qquad \forall k \in K, \forall (i,j) \in A, i \notin L \tag{6.14}$$

$$x_{il}^k = 1 \Rightarrow drive_l^{k,before} \leq T^{interval} \qquad \forall k \in K, \forall l \in L, \forall i \in D \cup O_1 \tag{6.15}$$

$$\sum_{i \in D} x_{ij}^k = y^k \qquad \forall k \in K, \forall j \in L \tag{6.16}$$

$$drive_l^{k,after} = 0 \qquad \forall k \in K, \forall l \in L \tag{6.17}$$

$$x_{ij}^k = 1 \Rightarrow drive_j^{k,after} \geq t_{ij}^k + drive_i^{k,after} \qquad \forall k \in K, \forall (i,j) \in A, j \notin L \tag{6.18}$$

$$drive_{o_2(k)}^{k,after} \leq T^{interval} \qquad \forall k \in K \tag{6.19}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, \forall k \in K \tag{6.20}$$

$$y_k \in \{0,1\} \qquad \forall k \in K \tag{6.21}$$

$$h_i \in \mathbb{R}^+ \qquad \forall i \in V \tag{6.22}$$

$$drive_i^{k,before} \in \mathbb{R} \qquad \forall i \in V, \forall k \in K \tag{6.23}$$

$$drive_i^{k,after} \in \mathbb{R} \qquad \forall i \in V, \forall k \in K \tag{6.24}$$

$$z_{ij}^\pi \in \{0,1\} \qquad \forall \pi \in \Pi, \forall i \in V_\pi S, \forall j \in V_\pi \tag{6.25}$$

La fonction objectif (6.3) est composée de trois termes minimisant (i) le coût d'utilisation des véhicules, (ii) le coût de transports et (iii) le coût horaire. Les contraintes (6.4) et (6.5) lient les variables x_{ij}^k et y^k et assurent que chaque véhicule commence sa tournée à son dépôt d'origine et termine sa tournée à son dépôt d'arrivée. La conservation des flots est assurée par les contraintes (6.6). Les contraintes (6.7) imposent le service de toutes les requêtes. Les contraintes (6.8) assurent la continuité de l'heure de début de service entre deux sommets successeurs et les contraintes (6.9) le respect des fenêtres de temps. Le respect des heures de début de service sur les ressources est assuré par les contraintes (6.10). Les contraintes (6.11) imposent la non simultanéité de deux tâches sur une ressource. Le temps maximum de conduite est imposé par les contraintes (6.12).

Les contraintes (6.13) (respectivement (6.17)) initialisent les variables de temps de conduite cumulés avant la pause (resp. après la pause-déjeuner). Les contraintes (6.14) (respectivement (6.18)) lient pour deux successeurs, les variables x_{ij}^k et $drive_j^{k,before}$ (resp. les variables x_{ij}^k et $drive_j^{k,after}$). Le respect des temps de conduite avant la pause-déjeuner (respectivement après la pause-déjeuner) est imposé par les contraintes (6.15) (resp. (6.19)). Les contraintes (6.16) assurent que chaque tournée utilisée passe par une pause-déjeuner.

Pour une meilleure compréhension du modèle mathématique, la modélisation présentée n'est pas linéaire à cause du produit de variables ou d'implications. Des techniques de linéarisation [102] peuvent être appliquées pour rendre cette modélisation linéaire.

Les contraintes (6.8) peuvent être remplacées par :

$$h_i + s_i + t_{ij} - h_j \leq (1 - x_{ij}^k) * M \qquad \forall (i,j) \in A, \forall k \in K \tag{6.26}$$

Les contraintes (6.10) peuvent être remplacées par :

$$h_i + s_i - h_j \leq (1 - z_{ij}^\pi) * M \qquad \forall \pi \in \Pi, \forall i \in V_\pi, \forall j \in V_\pi \tag{6.27}$$

Les contraintes (6.14) peuvent être remplacées par :

$$t_{ij}^k + drive_i^{k,before} - drive_j^{k,before} \leq (1 - x_{ij}^k) * M \qquad \forall k \in K, \forall (i,j) \in A \tag{6.28}$$

Les contraintes (6.15) peuvent être remplacées par :

$$drive_i^{k,before} - T^{interval} \leq (1 - x_{ij}^k) * M \qquad \forall k \in K, \forall j \in L, \forall i \in D \cup O_1 \tag{6.29}$$

Les contraintes (6.17) peuvent être remplacées par :

$$drive_j^{k,after} \leq (1 - x_{ij}^k) * M \qquad \forall k \in K, \forall i \in L, \forall j \in P \cup O_2 \tag{6.30}$$

Les contraintes (6.18) peuvent être remplacées par :

$$t_{ij}^k + drive_i^{k,after} - drive_j^{k,after} \leq (1 - x_{ij}^k) * M \qquad \forall k \in K, \forall (i,j) \in A \tag{6.31}$$

Dans la suite de cette partie, nous présentons la méthode de résolution de la deuxième phase du problème : le problème de tournées de véhicules. Pour cela, nous proposons de résoudre le problème en deux temps. Le chapitre 7 est consacré à la résolution d'une version simplifiée du problème, appelée FT-PDP-RS. Le chapitre 8 présente l'intégration des deux aspects industriels du problème au précédent algorithme : la minimisation des durées des tournées et la planification des pauses-déjeuner.

7

Résolution du FT-PDP-RS

Sommaire

Dans ce chapitre, nous nous intéressons à la résolution du problème $\mathcal{P}_2$ défini en introduction de la partie III. Dans un premier temps, nous présentons la formulation du problème avec les contraintes de synchronisation aux ressources (FT-PDP-RS). Cette formulation est présentée dans la section 7.1. Nous proposons une métaheuristique de type ALNS pour la résolution du FT-PDP-RS dans la section 7.2. Nous décrivons une nouvelle méthode pour tester la réalisabilité des insertions dans la section 7.3 et expliquons l'ordonnancement d'une solution dans la section 7.4. Des tests numériques sont présentés dans la dernière section 7.5. Nous concluons dans la section 7.6.

7.1 Le problème FT-PDP-RS

Avant de résoudre le Rich-FT-PDP-RS, nous nous proposons de résoudre une première version simplifiée où les contraintes liées à la gestion des chauffeurs ne sont pas prises en compte. Cette version est appelée FT-PDP-RS et correspond à un problème très proche de certaines applications de la littérature. Cette modélisation du problème permettra de comparer notre algorithme avec ceux de la littérature dans le but d'évaluer sa performance.

Le problème FT-PDP-RS est similaire à un PDPTW avec des transports uniquement effectués en camions complets. Les opérations de collectes (ensemble P) et de livraisons (ensemble D) sont effectuées sur des ressources (ensemble Π). À chaque instant, une et une seule opération peut être planifiée sur une ressource.

Le modèle mathématique du problème FT-PDP-RS est décrit de la manière suivante :

$$f = \min \sum_{k \in K} \mathcal{CD}^k * y^k + \sum_{k \in K} \sum_{(i,j) \in A} (\mathcal{CK}^k * x_{ij}^k * d_{ij}^k + \mathcal{CH}^k * x_{ij}^k * t_{ij}^k) + \sum_{i \in P} s_i * (\sum_{k \in K} \sum_{j \in D} \mathcal{CH}^k * x_{ij}^k) \tag{7.1}$$

$s.t.$

$$\sum_{(o_1(k),i) \in A} x_{o_1(k)i}^k = \sum_{(i,(o_2(k)) \in A} x_{io_2(k)}^k \qquad \forall k \in K \tag{7.2}$$

$$\sum_{(o_1(k),i) \in A} x_{o_1(k)i}^k = y^k \qquad \forall k \in K, i \neq o_2(k) \tag{7.3}$$

$$\sum_{(i,j) \in A} x_{ij}^k = \sum_{(i,j) \in A} x_{ji}^k \qquad \forall k \in K \tag{7.4}$$

$$\sum_{k \in K^r} x_{p_r d_r}^k = 1 \qquad \forall r \in R \tag{7.5}$$

$$h_j \geq (h_i + s_i + t_{ij}^k) x_{ij}^k \qquad \forall (i,j) \in A, \forall k \in K \tag{7.6}$$

$$e_i \leq h_i \leq l_i \qquad \forall i \in V \tag{7.7}$$

$$h_j \geq (h_i + s_i) z_{ij}^\pi \qquad \forall \pi \in \Pi, \forall i \in V_\pi, \forall j \in V_\pi \tag{7.8}$$

$$z_{ij}^\pi + z_{ji}^\pi = 1 \qquad \forall \pi \in \Pi, \forall i \in V_\pi, \forall j \in V_\pi \tag{7.9}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, \forall k \in K \tag{7.10}$$

$$y_k \in \{0,1\} \qquad \forall k \in K \tag{7.11}$$

$$h_i \in \mathbb{R}^+ \qquad \forall i \in V \tag{7.12}$$

$$z_{ij}^\pi \in \{0,1\} \qquad \forall \pi \in \Pi, \forall i \in V_\pi S, \forall j \in V_\pi \tag{7.13}$$

La modélisation de ce problème est identique à celle de $\mathcal{P}_2$ après avoir retiré les contraintes (6.12) à (6.19). La fonction objectif 7.1 f est différente et minimise la somme (i) des coûts fixe d'utilisation des véhicules, (ii) des coûts kilométriques et (iii) des coûts horaires des trajets et du service des demandes.

7.2 ALNS pour le FT-PDP-RS

Pour résoudre ce problème, nous proposons une méthode de recherche à voisinage large Large Neighborhood Search (LNS). Cette méthode, initialement introduite pour des problèmes de *Job-Shop* [2], a été présentée par Shaw [99] pour un problème de tournées de véhicules. Elle consiste à itérativement détruire, dans un premier temps, une partie d'une solution pour la réparer, dans un second temps. Elle est basée sur une recherche à voisinages larges par opposition à une méthode de recherche locale. Elle présente donc un

avantage dans les problèmes fortement contraints, tel que le problème de collecte et livraison, où le retrait et la réinsertion d'une requête (associant un point de collecte et un point de livraison) sont plus aisés que dans une simple recherche locale, notamment à cause de la dépendance temporelle entre le point de collecte et le point de livraison. C'est un constat qui est établi par Kindervarter et Savelsbergh [61]. Dans Shaw [99], les clients sont retirés suivant une mesure de proximité entre les clients. Ce principe sera étendu par Schrimpf avec la méthode de destruction et réparation Ruin and Recreate (R&R). De multiples façons de retirer les requêtes sont proposées. Le terme "opérateur" est utilisé pour désigner ces différentes heuristiques.

Pour assurer la convergence de la méthode, donc la décroissance de la fonction objectif dans un problème de minimisation, Schrimpf et al. [97] utilisent le critère du recuit simulé Simulated Annealing (SA) [62] ou Treshold Accepting (TA) [37].

La méthode LNS considère un opérateur de destruction à voisinage large et une méthode de réparation. Dans l'objectif d'utiliser plusieurs opérateurs de destruction et de réparation, Ropke et Pisinger [87] proposent une méthode *adaptative* pour sélectionner les opérateurs à chaque itération de l'algorithme. Leur méthode s'appelle méthode adaptative à voisinage large Adaptive Large Neighborhood Search (ALNS). La sélection se fait en utilisant une méthode de roulette biaisée pour choisir, à chaque itération, un opérateur de destruction et un opérateur de réparation en fonction de scores de performance calculés suivant le succès des opérateurs lors des dernières itérations.

L'algorithme 1 présente la description de cette méthode. À chaque itération de l'algorithme, la solution courante $s_{current}$ est détruite par un opérateur de destruction de l'ensemble $\mathcal{N}^-$ et est ensuite réparée par un opérateur de réparation de l'ensemble $\mathcal{N}^+$. Ces opérateurs sont choisis en fonction de leurs scores calculés par rapport à leur fonctionnement. A chaque itération, un nombre q de requêtes de l'ensemble R à retirer est choisi aléatoirement dans un intervalle $[\xi_{min} * |R|, \xi_{max} * |R|]$, ξ_{min} et ξ_{max} désignant un pourcentage minimum et maximum de retrait. Un critère d'acceptation sauvegarde la nouvelle solution courante $s_{current}$ et les scores des opérateurs utilisés sont mis à jour. La fonction f désigne la fonction objectif de notre problème.

7.2.1 Fonctions objectifs de l'ALNS

L'algorithme ALNS fonctionne avec deux fonctions objectifs distinctes. La fonction objectif standard est notée f et correspond à la fonction objectif du problème présenté dans l'équation 7.1.

Lors de l'exploration de l'ALNS, l'algorithme peut explorer des solutions irréalisables au sens où certaines requêtes ne seraient pas intégrées. Nous introduisons une variable binaire u_r qui vaut 1 si la requête est insérée et 0 autrement. Le coût d'une telle solution est donnée par une fonction objectif dite *pénalisée* représentée, f_p par l'équation 7.14.

$$\begin{aligned} f_p = \min & \sum_{k \in K} \mathcal{CD}^k * y^k \\ & + \sum_{k \in K} \sum_{(i,j) \in A} (\mathcal{CK}^k * x_{ij}^k * d_{ij}^k + \mathcal{CH}^k * x_{ij}^k * t_{ij}^k) \\ & + \sum_{i \in P} s_i * (\sum_{k \in K} \sum_{j \in D} \mathcal{CH}^k * x_{ij}^k) \\ & + \kappa \sum_{k \in K} \sum_{r \in R} (1 - u_r^k) \end{aligned} \tag{7.14}$$

Cette fonction objectif *pénalisée* autorise des requêtes à ne pas être insérées dans la solution. Le paramètre κ définit la pénalité pour les requêtes non insérées.

Algorithme 1 : STRUCTURE DE L'ALNS

Entrées : une solution initiale $s_{initial}$,un ensemble d'opérateurs de destruction $\mathcal{N}^-$, un ensemble d'opérateurs de réparation $\mathcal{N}^+$

Sortie : **retourner** la meilleure solution s_{best}

début
  $s_{best} \leftarrow s_{initial}$
  $s_{current} \leftarrow s_{initial}$
  tant que *critère d'arrêt non respecté* **faire**
    /* Sélection d'un opérateur de destruction */
    $opDestruction \leftarrow \text{chooseRemovalOperator}(\mathcal{N}^-)$
    /* Sélection d'un opérateur de réparation */
    $opReparation \leftarrow \text{chooseRepairOperator}(\mathcal{N}^+)$
    $s_{new} \leftarrow opDestruction(s_{current})$
    $s_{new} \leftarrow opReparation(s_{new})$
    si *transition est acceptée* **alors**
      $s_{current} \leftarrow s_{new}$
    fin
    si $f_p(s_{new}) < f_p(s_{best})$ **alors**
      $s_{best} \leftarrow s_{new}$
    fin
    $\text{updateScoreOperator}(\mathcal{N}^-,\mathcal{N}^+)$
  fin
  retourner s_{best}
fin

7.2.2 Aspect adaptatif

Par rapport à l'algorithme LNS, la méthode ALNS permet de choisir un opérateur de destruction et de réparation parmi tous les opérateurs. Pour cela, elle se base sur un score pour chaque opérateur qui est calculé selon son comportement sur les précédentes itérations.

La recherche de solutions de l'ALNS est divisée en segments : chaque segment représente un certain nombre d'itérations de l'ALNS. Dans chaque segment u, un opérateur i possède un poids $\mathrm{w}_{i,u}$. Ce poids est recalculé à la fin de chaque segment u pour le segment suivant $u+1$ avec la formule suivante :

$$\mathrm{w}_{i,u+1} = \mathrm{w}_{i,u}(1-r) + r\frac{\pi_i}{\theta_i}$$

r est un facteur de réaction qui contrôle l'ajustement des poids entre chaque segment. π_i est le score de l'opérateur i durant le segment en cours et θ_i est le nombre de fois où l'opérateur i a été choisi durant le dernier segment.

Le poids initial de chaque opérateur est 1. Dans le cas où il y a k opérateurs, à chaque segment temporel u, chaque opérateur i est pondéré avec un poids $\mathrm{w}_{i,u}$ dont la probabilité de sélection est la suivante :

$$\frac{\mathrm{w}_{i,u}}{\sum_{j=1}^{k} \mathrm{w}_{j,u}}$$

.

Le score de l'opérateur i sélectionné (destruction ou réparation) est ajusté à chaque itération en ajoutant un paramètre quantitatif $\sigma_1, \sigma_2, \sigma_3$ au score π_i de l'opérateur. Ces paramètres sont issus des travaux de Ropke et Pisinger [87] et présentés dans le tableau 7.1.

Paramètre	Description
σ_1	l'opérateur participe à la découverte d'une solution meilleure que la meilleure solution actuelle s_{best}
σ_2	l'opérateur participe à la découverte d'une solution meilleure que la solution courante actuelle $s_{current}$
σ_3	l'opérateur participe à la découverte d'une solution moins bonne que la solution courante actuelle $s_{current}$ mais toutefois acceptée par le critère d'acceptation

Tableau 7.1 – Description des paramètres utilisés pour quantifier le succès des opérateurs.

Le critère d'acceptation d'une solution courante est le recuit simulé, présenté dans la section suivante. En fonction du comportement de ce critère, nous récompensons de façons différentes la découverte d'une nouvelle solution courante $s_{current}$:

- une solution meilleure que la solution courante est trouvée : récompense avec σ_2 ;
- une solution moins bonne que la solution courante, mais acceptée par le critère d'acceptation, est trouvée : récompense avec σ_3.

Ces paramètres sont ajoutés de manière cumulative en fonction de la réalisation de chacun des cas. Si un opérateur participe à plusieurs des réalisations citées précédemment, son score est récompensé par la sommes des différents paramètres.

7.2.3 Critère d'acceptation d'une solution

Pour que la méthode ALNS converge vers un minimum sans toutefois être bloquée dans un minimum local, nous choisissons d'utiliser la méthode SA comme critère d'acceptation. Cette méthode a été proposée initialement par Kirkpatrick et al. [62] et est inspirée du procédé métallurgique du recuit. Elle permet à un

algorithme de converger tout en offrant la possibilité d'explorer des solutions de moins bonne qualité lors de la recherche. Ropke et Pisinger [87] utilisent cette méthode dans leurs travaux sur l'ALNS.

Soit T la température de l'ALNS à une itération. Une solution nouvelle s_{new} est acceptée comme solution courante $s_{current}$ avec la probabilité suivante $e^{-\frac{\Delta}{\mathrm{T}}}$ où $\Delta = f_p(s_{new}) - f_p(s_{current})$ avec f_p la fonction objectif pénalisée.

La température initiale du recuit simulé est T_{start}. À chaque itération, la température décroit selon une vitesse de refroidissement c en utilisant la formule $\mathrm{T} = \mathrm{T} * \mathrm{c}$.

La température initiale est calculée suivant la méthode décrite par Ropke et Pisinger [87]. La température initiale est calculée de façon qu'une solution qui est $\mathtt{w}$ (valeur exprimée en pourcentage) pire que la solution courante soit acceptée avec une probabilité de 0.5. Nous prendrons la solution initiale $s_{initial}$ pour calculer la température initiale.

$$
\begin{aligned}
e^{-\frac{(1+\mathtt{w})*f_p(s_{initial})-f_p(s_{initial})}{\mathrm{T}_{start}}} &= 0.5 \\
e^{-\frac{\mathtt{w}*f_p(s_{initial})}{\mathrm{T}_{start}}} &= 0.5 \\
\mathrm{T}_{start} &= \frac{\mathtt{w} * f_p(s_{initial})}{-\ln(0.5)}
\end{aligned}
$$

7.2.4 Modélisation d'une solution

Une solution de notre problème est constitué d'un ensemble de tournées. Dans chaque tournée, des requêtes sont servies successivement. Les points de collecte et de livraison de ces requêtes peuvent être affectés à une ressource. Pour modéliser les opérateurs ainsi que les calculs que nous effectuons sur cette solution, nous introduisons un graphe temporel G_t qui inclue tous les sommets de FT-PDP-RS. L'ensemble des sommets de G_t contient tous les sommets de G.

Chaque précédence entre deux sommets successifs dans une tournée ou sur une ressource est modélisée comme un arc dans G_t. Le poids de l'arc entre deux sommets i et j est le temps minimum entre l'heure de service à i et à j. Le successeur direct d'un sommet i est noté ρ_i et son prédécesseur direct σ_i. L'ensemble des successeurs d'un sommet i est noté $\Gamma(i)$.

La méthode $\mathrm{enleverRequête}$ est une méthode qui enlève une requête r d'une solution. Concrètement, le point de collecte p_r et le point de livraison d_r sont déconnectés du graphe temporel. En définissant ρ_i le prédécesseur et σ_i le successeur d'un point i, le retrait de la requête r revient à supprimer les arcs avec les prédécesseurs et les successeurs sur les tournées et les ressources et à recréer les arcs entre le successeur et le prédécesseur de chaque sommet enlevé. La figure 7.1 illustre le retrait d'une requête r. À l'issue de ce retrait, les sommets de collecte et livraison restent connectés entre eux.

La méthode $\mathrm{RequestInsertion}$ est une méthode qui ajoute une requête r dans une solution. Concrètement, le point de collecte p_r et le point de livraison d_r sont connectés entre le point i et le point σ_i. La méthode $\mathrm{ResourceScheduling}$ connecte le point de collecte p_r sur la ressource de collecte entre les sommets u et σ_u et le point de livraison d_r sur la ressource de livraison entre les sommets v et σ_v. L'ajout de cette requête est représenté par la figure 7.2.

La méthode $\mathrm{updateEarliest}$ permet de recalculer le graphe temporel d'une solution avec un ordonnancement au plus tôt. Cette méthode sera présentée ultérieurement dans la section 7.4. Dans chaque opérateur de destruction ou de réparation présenté, nous considérons un ordonnancement au plus tôt du graphe G_t, ce qui revient à calculer l'ordonnancement dans un diagramme PERT.

Cette modélisation implique le choix d'un ordre sur chaque ressource. Comme la figure 7.2 le montre sur la ressource U, il existe une précédence entre le sommet u et le sommet p_r et une précédence entre le sommet p_r et le sommet σ_u.

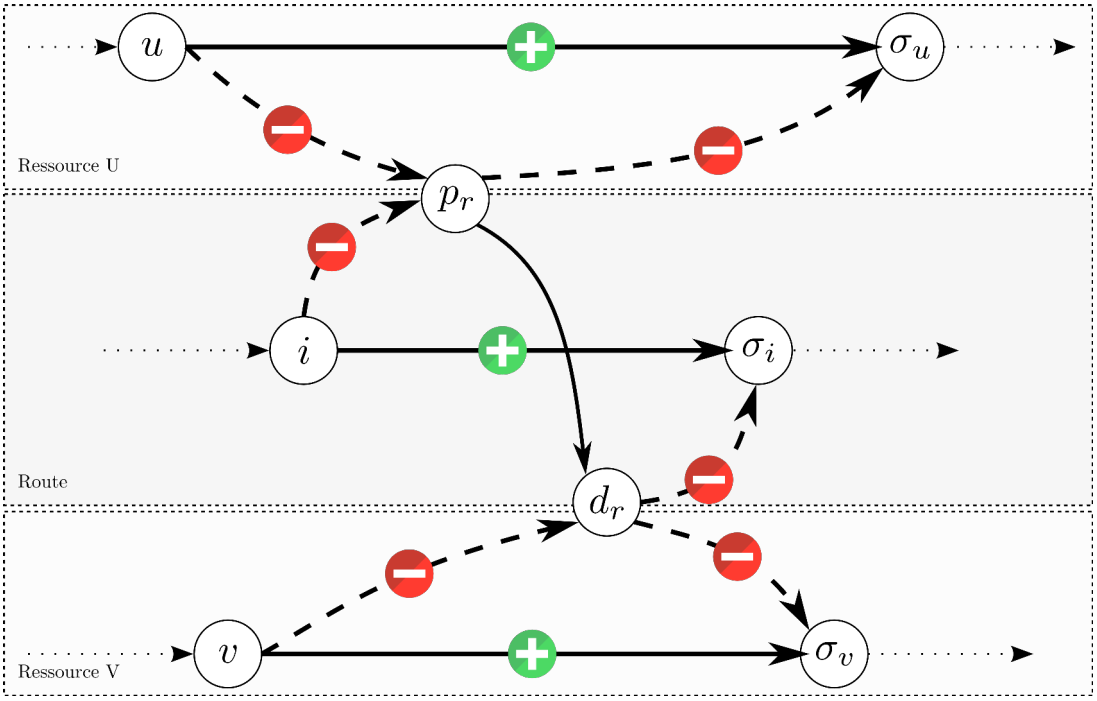


FIGURE 7.1 – Retrait d'une requête r. Les arcs retirés sont symbolisés par le symbole ⛔ et les arcs ajoutés par le symbole ⊕.

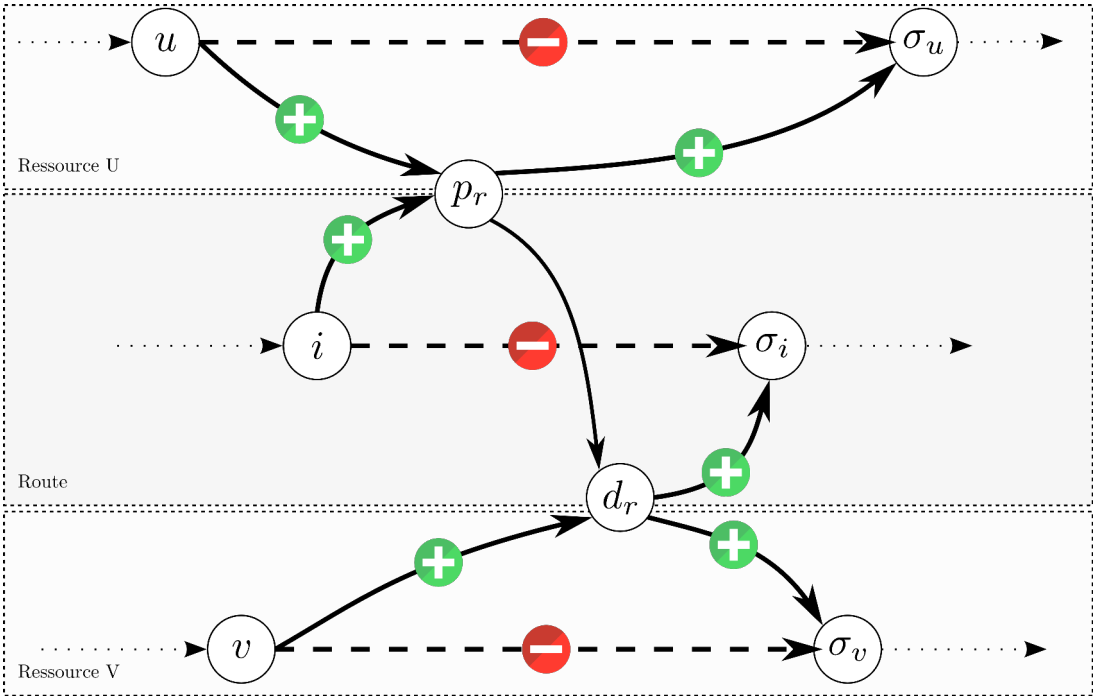


FIGURE 7.2 – Ajout d'une requête r. Les arcs retirés sont symbolisés par le symbole ⛔ et les arcs ajoutés par le symbole ⊕.

7.2.5 Opérateurs de destruction

Nous présentons dans cette partie les opérateurs de destruction utilisés dans l'algorithme ALNS. Nous définissons plusieurs ensembles pour comprendre le fonctionnement des opérateurs. L'ensemble des requêtes non insérées dans une solution est $\mathcal{R}$ et l'ensemble des requêtes insérées dans cette solution est $\mathcal{L}$.

Une solution partielle $s_{partial}$ est une solution dans laquelle un nombre de requêtes a été retiré. Pour faciliter la compréhension du problème, nous proposons de détailler deux méthodes qui sont récurrentes dans la description des opérateurs de destructions.

Chaque opérateur de destruction est présenté sous sa forme anglaise avec son abréviation. Nous définissons son fonctionnement dans chacun des cas.

RDO - Random Destroy Operator

L'opérateur de destruction **RDO** retire un nombre q de requêtes aléatoirement. À chaque itération, la requête est enlevée en utilisant la méthode $\mathrm{enleverRequête}$. À la fin du retrait des q requêtes, la solution est mise à jour en effectuant un calcul du graphe temporel en utilisant la méthode $\mathrm{updateEarliest}$.

Algorithme 2 : RDO - RANDOM DESTROY OPERATOR

Entrées : une solution initiale $s_{initial}$, un nombre de requêtes à enlever q
Sortie : une solution détruite $s_{partial}$

début
  $s_{partial} \leftarrow s_{initial}$
  tant que $q > 0$ **faire**
    /* Choix d'une requête r à retirer */
    $r \leftarrow \mathrm{choixAléatoire}(s_{initial})$
    /* Enlever la requête r de la solution */
    $s_{partial} \leftarrow \mathcal{L} * y^{randomness}$
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$
    $\mathcal{L} \leftarrow \mathcal{L} \setminus \{r\}$
    $q \leftarrow q - 1$
  fin
  $\mathrm{updateEarliest}(s_{partial})$
  retourner $s_{partial}$
fin

WDO - Worst Destroy Operator

L'opérateur de destruction **WDO** retire un nombre q de requêtes de coût élevé. Nous définissons par $\Delta(r, s)$ la différence de coût d'une solution s entre son coût original $f(s)$ et son coût sans la requête r. À chaque itération, les requêtes de $\mathcal{L}$ sont triées par ordre décroissant de coût de retrait Δ. Le coût de retrait $\Delta(r, s)$ d'une requête r, servie par un véhicule $k \in K$, d'une solution s est donné par la formule suivante :

$$\Delta(r, s) = \mathcal{CK}^k * (d_{\rho_{p_r} p_r} + d_{d_r \sigma_{d_r}} + d_{p_r d_r} - d_{\rho_{p_r} \sigma_{p_r}})$$

Un nombre aléatoire y est choisi aléatoirement dans l'intervalle $[0, 1]$ et est pondéré par un paramètre $randomness$ à caractère aléatoire donnant plus d'importance aux requêtes de plus fort coût de retrait. La requête à retirer r est choisie dans la liste $\mathcal{L}$. Plus le paramètre $randomness$ est élevé, plus les requêtes situées au début de la liste $\mathcal{L}$ ont de chance d'être sélectionnées. Une fois qu'une requête est choisie, elle est retirée du graphe temporel. Celui-ci est mis à jour à la fin du retrait des q requêtes en utilisant la méthode $\mathrm{updateEarliest}$.

Algorithme 3 : WDO - WORST DESTROY OPERATOR

Entrées : une solution initiale $s_{initial}$, un nombre de requêtes à enlever q
Sortie : une solution détruite $s_{partial}$

début
  $s_{partial} \leftarrow s_{initial}$
  tant que $q > 0$ **faire**
    /* Trier les requêtes de $\mathcal{L}$ par coût décroisant $\Delta(r, s)$ */
    $\mathrm{L} \leftarrow \mathrm{trierCoutDescendantRetrait}(\mathcal{L})$
    /* Choix d'un nombre aléatoire $y \in [0, 1]$ */
    $y \leftarrow rand(0, 1)$
    /* Choix d'une requête r à retirer */
    $r \leftarrow \mathcal{L} * y^{randomness}$
    /* Enlever la requête r de la solution */
    $s_{partial} \leftarrow \mathrm{enleverRequête}(s_{partial})$
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$
    $\mathcal{L} \leftarrow \mathcal{L} \setminus \{r\}$
    $q \leftarrow q - 1$
  fin
  $\mathrm{updateEarliest}(s_{partial})$
  retourner $s_{partial}$
fin

DRDO - Distance-Related Destroy Operator

L'opérateur de destruction **DRDO** est un opérateur de retrait basé sur une mesure de distance entre les requêtes. L'idée générale a été initialement présentée par Shaw [98, 99] et a pour objectif de retirer des requêtes qui sont *proches* dans l'idée de les insérer d'une meilleure façon par la suite.

Pour chaque requête i, on calcule une mesure de distance r_{ij}^{dist} avec les autres requêtes $j \neq i$. Nous définissons cette mesure de distance ainsi :

$$r_{ij}^{dist} = \frac{1}{2}(d_{p_i d_j} + d_{d_i p_j})$$

L'opérateur **DRDO** choisit premièrement une requête $i \in \mathcal{L}$ de manière aléatoire qu'il retire de la solution et que l'on met dans une liste temporaire $\mathcal{D}$. Les requêtes restantes de $j \in \mathcal{L}$ sont ordonnées par ordre croissant suivant la distance par rapport à la requête i. Cette mesure reflète l'aspect "camions pleins" de notre problème.

Un nombre aléatoire y est choisi aléatoirement dans l'intervalle $[0, 1]$ et est pondéré par un paramètre $randomness$ à caractère aléatoire donnant plus d'importance aux requêtes avec la plus faible distance. La requête r est enlevée de $\mathcal{L}$ et est rajoutée à $\mathcal{D}$.

Ce processus est réitéré, en partant d'une requête i choisie aléatoirement dans $\mathcal{D}$, jusqu'à ce qu'il y ait q requêtes retirées.

TRDO - Time-Related Destroy Operator

L'opérateur de destruction **TRDO** fonctionne de la même manière que l'opérateur **DRDO**. Cependant, la requête initiale sélectionnée i pour calculer la distance est la même durant tout l'opérateur. Pour une requête i, on évalue une mesure de distance r_{ij}^{time} avec les autres requêtes $j \neq i$. Nous définissons cette mesure de distance ainsi :

Algorithme 4 : DRDO - DISTANCE-RELATED DESTROY OPERATOR

Entrées : une solution initiale $s_{initial}$, un nombre de requêtes à enlever q
Sortie : une solution détruite $s_{partial}$

début
  $s_{partial} \leftarrow s_{initial}$
  /* Choisir une aléatoirement requête i */
  $i \leftarrow$ choixRequête($s_{initial}$)
  D $= \{i\}$
  tant que $q > 0$ **faire**
    /* Choisir une requête aléatoirement dans D */
    $i \leftarrow$ choixRequêteAléatoire(D)
    /* Trier par ordre croissant les requêtes $j \in \mathcal{L}$ par mesure de distance r_{ij}^{dist} */
    L $\leftarrow$ trierMesureDistance($\mathcal{L}$)
    /* Choix d'un nombre aléatoire $y \in [0, 1]$ */
    $y \leftarrow rand(0, 1)$
    /* Choix d'une requête j à retirer */
    $j \leftarrow \mathcal{L} * y^{randomness}$
    D $\leftarrow$ D $\cup \{j\}$
    $\mathcal{L} \leftarrow \mathcal{L} \setminus \{j\}$
    $q \leftarrow q - 1$
  fin
  /* Retirer les requêtes de la solution */
  pour chaque *requête* $r \in$ D **faire**
    $s_{partial} \leftarrow$ enleverRequête(r)
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$
  fin
  updateEarliest($s_{partial}$)
  retourner $s_{partial}$
fin

$$r_{ij}^{time} = |h_{p_i} - h_{p_j}| + |h_{d_i} - h_{d_j}|$$

Cette requête permet de retirer des requêtes étant servies dans les mêmes plages horaires que la requête initiale.

RODO - Route Destroy Operator

L'opérateur de destruction **RODO** est un opérateur qui enlève toutes les requêtes d'une tournée. Une tournée est sélectionnée aléatoirement et toutes les requêtes de cette tournée sont retirées. Le procédé est répété jusqu'à ce qu'il y ait au moins q requêtes retirées.

Algorithme 5 : RODO - ROUTE DESTROY OPERATOR

Entrées : une solution initiale $s_{initial}$, un nombre de requêtes à enlever q
Sortie : une solution détruite $s_{partial}$

début
  $s_{partial} \leftarrow s_{initial}$
  tant que $q > 0$ **faire**
    /* Choisir une tournée k aléatoirement */
    pour chaque *requête* $r \in R_k$ **faire**
      /* Retirer les requêtes de la solution */
      $s_{partial} \leftarrow \text{enleverRequête}(r)$
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$
      $\mathcal{L} \leftarrow \mathcal{L} \setminus \{r\}$
      $q \leftarrow q - 1$
    fin
  fin
  $\text{updateEarliest}(s_{partial})$
  retourner $s_{partial}$
fin

REDO - Resource Destroy Operator

L'opérateur de destruction **REDO** est un opérateur qui enlève toutes les requêtes d'une ressource. Une ressource est sélectionnée aléatoirement et toutes les requêtes de cette ressource sont retirées. Le procédé est répété jusqu'à ce qu'il y ait au moins q requêtes retirées. La solution est mise à jour une fois que toutes les requêtes ont été retirées.

CDO - Critical Destroy Operator

L'opérateur de destruction **CDO** est un opérateur qui enlève les requêtes d'une solution par rapport à leur *forward time slack*. Les *forward time slack* sont détaillés plus précisément dans la partie 7.3. Le *forward time slack* d'un sommet i, noté F_i, est la marge (en unité temporelle) dont peut être décalée l'heure de début de service de ce sommet dans un ordonnancement au plus tôt. Le *forward time slack* garantit que les sommets successeurs de i n'auront pas de fenêtres de temps violées lors du décalage de i.

Les requêtes sont triées par ordre croissant de criticité qui est définie, pour une requête r par $\min(F_{p_r}, F_{d_r})$.

Un nombre aléatoire y est choisi aléatoirement dans l'intervalle $[0, 1]$ et est pondéré par un paramètre $randomness$ à caractère aléatoire donnant plus d'importance aux requêtes avec la plus faible criticité. La solution est mise à jour une fois que toutes les requêtes ont été retirées avec la méthode updateEarliest.

Algorithme 6 : REDO - RESOURCE DESTROY OPERATOR

Entrées : une solution initiale $s_{initial}$, un nombre de requêtes à enlever q
Sortie : une solution détruite $s_{partial}$

début
  $s_{partial} \leftarrow s_{initial}$
  tant que $q > 0$ **faire**
    /* Choisir une ressource π aléatoirement */
    pour chaque *requête* $r \in R_\pi$ **faire**
      /* Retirer les requêtes de la solution */
      $s_{partial} \leftarrow$ enleverRequête(r)
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$
      $\mathcal{L} \leftarrow \mathcal{L} \setminus \{r\}$
      $q \leftarrow q - 1$
    fin
  fin
  updateEarliest$(s_{partial})$
  retourner $s_{partial}$
fin

Algorithme 7 : CDO - CRITICAL DESTROY OPERATOR

Entrées : une solution initiale $s_{initial}$, un nombre de requêtes à enlever q
Sortie : une solution détruite $s_{partial}$

début
  $s_{partial} \leftarrow s_{initial}$
  tant que $q > 0$ **faire**
    /* Trier les requêtes de $\mathcal{L}$ par criticité ascendante */
    L $\leftarrow$ trierParCriticite$(\mathcal{L})$
    /* Choix d'un nombre aléatoire $y \in [0, 1]$ */
    $y \leftarrow rand(0, 1)$
    /* Choix d'une requête r à retirer */
    $r \leftarrow \mathcal{L} * y^{randomness}$
    /* Enlever la requête r de la solution */
    $s_{partial} \leftarrow$ enleverRequête$(s_{partial})$
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$
    $\mathcal{L} \leftarrow \mathcal{L} \setminus \{r\}$
    $q \leftarrow q - 1$
  fin
  updateEarliest$(s_{partial})$
  retourner $s_{partial}$
fin

Cet opérateur est motivé dans notre problème par les contraintes de synchronisation aux ressources qui impliquent des marges de temps faibles sur certains sommets, ce qui a tendance à contraindre la solution. L'objectif est de retirer les requêtes les plus contraintes.

7.2.6 Opérateurs de réparation

Pour réparer une solution, nous devons réinsérer les requêtes de l'ensemble $\mathcal{R}$ dans les tournées et, si nécessaire, dans les ressources d'une solution.

Un opérateur de réparation doit choisir la requête à insérer, la tournée et la position dans cette tournée ainsi que la ressource et la position sur cette ressource (à la fois pour le point de collecte et le point de livraison).

À la différence des opérateurs de réparation de la littérature pour l'ALNS qui utilisent le critère d'insertion sur la tournée, nous devons, dans notre cas, décider d'un ordonnancement à la fois sur la tournée et la ressource. Pour cela, nous proposons un *framework* général qui permettra de définir les différents opérateurs de réparation. Ce *framework* est basé sur les travaux de Ebben et al. [41] qui utilisent des règles de priorité pour chaque type d'insertion.

Dans la suite de cette section, nous définissons les différentes règles utilisées pour la définition des opérateurs de réparation.

Framework général des opérateurs de réparation

Le *framework* proposé combine des heuristiques classiques de réparation pour le VRPTW introduites par Solomon [100] et des heuristiques d'ordonnancement introduites par Kolisch [63]. Trois décisions sont à prendre :

1. sélection d'une requête (méthode $\mathrm{RequestSelection}$) ;
2. insertion de la requête dans une tournée (méthode $\mathrm{RequestInsertion}$) ;
3. ordonnancement de la requête sur la ressource (méthode $\mathrm{ResourceScheduling}$).

Pour chaque décision, plusieurs règles sont considérées. Certaines sont inspirées de la littérature et d'autres ont été définies pour le problème. Les objectifs des décisions sont les suivants :

- $\mathrm{RequestSelection}$: détermine l'ordre dans lequel les requêtes sont sélectionnées pour l'insertion dans les tournées ;
- $\mathrm{RequestInsertion}$: détermine dans quelle tournée et dans quelle position dans cette tournée, la requête sélectionnée doit être insérée ;
- $\mathrm{ResourceScheduling}$: détermine à quelle position dans la ressource le point de collecte ou de livraison de la requête sélectionnée doit être inséré.

Nous considérons une solution partielle $s_{partial}$ dans laquelle des requêtes ne sont pas servies ($\mathcal{R} \neq \emptyset$). Le *framework* des opérateurs de réparation est proposé dans l'algorithme 8.

Nous détaillons, dans la suite de ce document, les règles considérées pour chaque décision.

Sélection d'une requête ($\mathrm{RequestSelection}$)

Dans cette partie, nous présentons cinq règles pour choisir la requête à insérer.

- Earliest Due Date (**EDD**) : sélectionne la requête dont la livraison doit être finie au plut tôt ($\arg\min_{r \in R} l_{d_r}$). Cette règle EDD est fréquemment utilisée dans les problèmes d'ordonnancement, voir [19].

Algorithme 8 : *Framework* OPÉRATEURS DE RÉPARATION

Entrées : une solution partielle $s_{partial}$, un ensemble de requêtes non insérées $\mathcal{R}$
Sortie : une solution complète

```
1 début
2     tant que plus d'insertion réalisable faire
          /* Sélection d'une requête à insérer                    */
3         r ← RequestSelection(R) ;
          /* Insertion de r dans la tournée                       */
4         s ← RequestInsertion(r) ;
          /* Insertion r dans la ressource                        */
5         ResourceScheduling(r) ;
6         R ← R − {r} ;
7     fin
8 fin
```

- Earliest Delivery Release Date (**EDRD**) : sélectionne la requête qui a la plus petite borne inférieure pour l'heure de service au point de livraison ($\arg\min_{r \in R} e_{d_r}$) ;
- Number of Similar Requests (**NSR**) : le paramètre NSR_r indique le nombre de requêtes provenant de la même demande que la requête r. Les requêtes avec le plus grand *NSR* sont sélectionnées en priorité.
- Cheapest Insertion (**CI**) : sélectionne la requête avec le meilleur coût d'insertion parmi toutes les positions parmi toutes les tournées.
- K-Regret (**KR**) : le regret c_r^k d'ordre $k > 2$ d'une requête r est défini ainsi : $c_r^k = \sum_{i=2}^{k} f_i - f_1$ avec f_1 le coût d'insertion de la requête r dans la meilleure tournée, f_i le coût d'insertion de la requête r dans la $i^{\text{ème}}$ meilleure tournée. La requête r avec la plus grande valeur de regret c_r^k est choisie. Pour notre problème, nous utiliserons le regret d'ordre 2 (**2R**) et d'ordre 3 (**3R**).
- Critical Resource (**CR**) : la stratégie de cette règle est de réordonnancer les requêtes d'une ressource occupée d'une meilleure façon. Les requêtes appartenant à la ressource la plus occupée sont sélectionnées en priorité.

Insertion d'une requête dans la tournée ($\mathrm{RequestInsertion}$**)**

Deux règles sont définies pour choisir la tournée et la position dans laquelle insérer la requête sélectionnée.

- Cheapest Insertion (**CI**) : insère la requête sélectionnée à la position qui génère la plus faible augmentation de la fonction objectif. Si deux insertions (i.e. deux coûts) sont identiques, la requête est insérée dans la tournée qui donne la plus petite heure de service pour la collecte de cette requête.
- Earliest Truck Appending (**ETA**) : insère la requête à la fin d'une tournée qui induit la plus petite heure de service de la collecte de la requête.

Ordonnancement de la requête sur la ressource ($\mathrm{ResourceScheduling}$**)**

Deux règles sont proposées pour déterminer la position du point de collecte et du point de livraison dans les ressources concernées.

- Earliest Feasible (**EF**) : insère la requête dans la ressource à la position réalisable au plus tôt ;
- Latest Feasible (**LF**) : insère la requête dans la ressource à la position réalisable au plus tard.

Bruitage des opérateurs d'insertion

Ropke et Pisinger [87] ont introduit le bruitage pour randomiser les insertions. Lorsqu'un opérateur de réparation est sélectionné, nous décidons aléatoirement d'utiliser ou non le bruitage en se basant sur le succès du bruitage sur les précédentes itérations. Pour un coût C d'insertion d'un opérateur de réparation, le coût d'insertion avec bruitage C' est défini par $C' = \max\{0, C + noise\}$. La quantité de bruit ajouté $noise$ est un nombre aléatoire choisi dans l'intervalle $[-0.025 * d_{max}, 0.025 * d_{max}]$ où d_{max} est la distance maximale entre deux sommets de V.

La valeur de 0.025 a été prise dans [87]. Les auteurs de cet article ont fixé cette valeur qui offre les meilleurs résultats.

7.3 Réalisabilité d'une insertion

L'ALNS est un algorithme qui, à chaque itération, enlève et ajoute des requêtes dans un graphe temporel. Un grand nombre d'insertions est évalué à chaque itération : premièrement vis à vis du coût d'insertion et deuxièmement vis à vis de la réalisabilité. Nous considérons toutes les insertions par coût croissant jusqu'à ce qu'une insertion réalisable soit trouvée.

En particulier, les précédences entre les points, sur les tournées et les ressources, impliquent de vérifier que les requêtes sont servies dans leurs fenêtres de temps. Pour tester si une solution est réalisable, des méthodes ont été proposées dans la littérature. Dans cette partie, nous rappelons quelques résultats de la littérature et nous proposons une extension à la méthode de réalisabilité, proposée par Masson et al. [71], spécialement pour notre problème FT-PDP-RS avec pour objectif de satisfaire la synchronisation sur les ressources.

7.3.1 Réalisabilité d'une insertion pour le TSPTW

Savelsbergh a introduit les *forward time slack* pour le TSPTW dans [92, 93]. Cette valeur représente le décalage temporel maximal que l'on peut faire à un sommet u sans que les fenêtres de temps des successeurs de u en soient violées. Dans le TSPTW, l'idée est d'effectuer la vérification des opérateurs de recherche locale en temps constant. Nous introduisons maintenant plusieurs notations. $\psi_{u,v}$ est l'ensemble ordonné des sommets sur le chemin $(u, \ldots, v)$ et $\Omega_{u,v}$ est l'ensemble de tous les chemins de u vers v. Nous appelons $TWT_{\psi_{u,v}} = \Sigma_{i \in \psi_{\sigma(u),v}} w_i$, le temps total d'attente sur le chemin $(u, \ldots, v)$. Cordeau et al. [25] ont défini le *forward time slack* F_u d'un sommet u par :

$$F_u = \min_{i \in \{u\} \cup \Gamma(u)} \Big\{ TWT_{\psi_{u,i}} + l_i - h_i \Big\},$$

où h_i est l'heure de service au sommet i dans un ordonnancement au plus tôt et l_i la borne supérieure de la fenêtre de temps du même sommet i.

7.3.2 Réalisabilité d'une insertion pour le PDPT

Dans le PDPT, différents chemins peuvent exister entre deux sommets du graphe à cause des contraintes de précédences présentes aux nœuds de transfert. Masson et al. [71] ont étendu la notion de *forward time slack* pour ce problème. Premièrement, la notion de graphe temporel est introduite. Ce graphe représente toutes les précédences entre les différents sommets du problème comme des arcs temporels. Pour chaque chemin $\omega \in \Omega_{u,v}$ dans ce graphe, TWT_ω représente la somme des temps d'attente dans le chemin ω dans un ordonnancement au plus tôt. Le *slack time* $ST_{u,v}$ entre deux sommets u et v est défini par :

$$ST_{u,v} = \min_{\omega \in \Omega_{u,v}} TWT_\omega.$$

S'il n'existe aucun chemin entre u et v, $ST_{u,v}$ est égal à $+\infty$. Cela correspond au cas où il n'y a pas de chemin via les tournées et les ressources entre les sommets u et v. Avec cette définition, on peut réécrire le *forward time slack* du sommet u donné par l'équation 7.15.

$$F_u = \min_{i \in \{u\} \cup \Gamma(u)} \left\{ ST_{u,i} + l_i - h_i \right\} \quad \forall k \in K \tag{7.15}$$

7.3.3 Réalisabilité d'une insertion pour le FT-PDP-RS

Pour assurer la réalisabilité de cette insertion, nous devons vérifier les deux conditions suivantes proposées dans [71] :

1. l'insertion ne crée pas de cycle dans le graphe temporel G_t ;
2. l'insertion ne viole aucune fenêtre de temps.

Détection de cycle

L'insertion de la requête crée de nouvelles précédences entre les sommets de la solution courante. Pour s'assurer de la réalisabilité de cette insertion, nous devons, dans un premier temps, tester si des cycles ont été introduits dans le graphe. Le tableau 7.2 représente l'ensemble minimum des nouvelles précédences créées par l'insertion représentée dans la figure 7.2. L'objectif est de tester si les nouvelles précédences ne sont pas en contradiction avec des précédences déjà existantes de la solution. Lorsqu'une insertion est réalisée, on teste que les précédences inverses n'existent pas : $i \in \Gamma(\sigma_u), i \in \Gamma(\sigma_v), u \in \Gamma(\sigma_v), u \in \Gamma(\sigma_i), v \in \Gamma(\sigma_i)$. Avec une représentation matricielle des successeurs (taille $|V| \times |V|$), renseignée dans une phase de pré-processing, on peut tester en temps constant si une insertion crée un cycle.

	σ_i	σ_u	σ_v
i		$\rightarrow$	$\rightarrow$
u	$\rightarrow$		$\rightarrow$
v	$\rightarrow$	$\rightarrow$	

Tableau 7.2 – Liste des nouvelles précédences créées par l'insertion représentée dans la figure 7.2.

Réalisabilité temporelle

La réalisabilité d'une insertion est évaluée par rapport aux contraintes temporelles, en testant si une fenêtre de temps est violée ou non. Nous notons $\bar{h}_i$ l'heure à laquelle le service au point i débute après l'insertion de la requête r. L'algorithme 9 présente la séquence de tests impliquant les variables de prétraitement et vérifiant la réalisabilité des contraintes temporelles du problème. Cet algorithme décrit le cas général avec une synchronisation sur la ressource de collecte et la ressource de livraison dans le cas où $\sigma_u \notin \Gamma(\sigma_i)$ et $\sigma_v \notin \Gamma(\sigma_i)$.

Dans le graphe temporel G_t, il est possible que le sommet σ_u soit un successeur de σ_i avec $\sigma_u \in \Gamma(\sigma_i)$. Nous avons présenté le cas général dans l'algorithme 9 mais nous devons considérer toutes les possibilités entre les sommets σ_u, σ_v et σ_i. Ceci conduit à evaluer les décalages dans l'ordre topologique de ces sommets. Les opérations sont les mêmes, uniquement l'ordre change.

L'algorithme 9 est une extension de l'algorithme proposé par Masson et al. [71] et a une complexité temporelle constante en $\mathcal{O}(1)$.

Ainsi, une insertion peut être évaluée en temps constant si les deux tests suivants sont réalisés : (i) toutes les heures de service sont calculées dans un ordonnancement au plus tôt, (ii) la valeur du *forward time slack* F_v pour tout $v \in G_t$ et la matrice des successeurs a été mise à jour après chaque destruction ou réparation.

Algorithme 9 : RÉALISABILITÉ D'UNE INSERTION DE LA REQUÊTE r

Entrées : une requête r à insérer après le sommet i avec le point de collecte p_r à insérer avant σ_u et le point de livraison d_r à insérer avant σ_v

Sortie : **retourner** VRAI si l'insertion est réalisable, FAUX autrement

début
  /* Insertion du sommet p_r */
  $\bar{h}_{p_r} = \max(e_{p_r}, h_i + s_i + t_{i,p_r}, h_u + s_u)$
  if $\bar{h}_{p_r} > l_{p_r}$ **then**
    return FAUX
  end
  /* Evaluer le décalage au sommet σ_u */
  $\delta_{\sigma_u} = \bar{h}_{p_r} + s_{p_r} - h_{\sigma_u}$
  if $\delta_{\sigma_u} > F_{\sigma_u}$ **then**
    return FAUX
  end
  /* Evaluer le début de service au sommet d_r */
  $\bar{h}_{d_r} = \max(e_{d_r}, \bar{h}_{p_r} + s_{p_r} + t_{p_r,d_r}, h_v + s_v)$
  if $\bar{h}_{d_r} > l_{d_r}$ **then**
    return FALSE
  end
  /* Evaluer le décalage au sommet σ_v */
  $\delta_{\sigma_v} = \bar{h}_{d_r} + s_{d_r} - h_{\sigma_v}$
  if $\delta_{\sigma_v} > F_{\sigma_v}$ **then**
    return FAUX
  end
  /* Evaluer le décalage au sommet σ_i */
  $\delta_{\sigma_i} = \bar{h}_{d_r} + s_{d_r} + t_{d_r,\sigma_i} - h_{\sigma_i}$
  if $\delta_{\sigma_i} > F_{\sigma_i}$ **then**
    return FAUX
  else
    return VRAI
  end
fin

Ce prétraitement réduit de manière significative la complexité et le temps de calcul des possibilités d'insertions lors de la phase de réparation.

7.4 Ordonnancement du graphe temporel d'une solution G_t

Les opérations effectuées sur la solution sont réalisées sur une solution qui est ordonnancée au plus tôt. Cet ordonnancement nous permet de calculer l'heure au plus tôt de service des différents sommets du graphe ainsi que les *forward time slack* associés.

Après chaque opération d'insertion ou après un ensemble d'opérations de retrait d'une requête r dans G_t, il faut donc procéder à un nouveau calcul du graphe pour connaître les nouvelles heures de service et *forward time slack* des sommets. La fonction $\mathrm{updateEarliest}$ réalise cette fonction et est présentée dans l'algorithme 10.

Il est important de détailler ici la complexité de cet algorithme car il sera appelé à chaque changement (ajout ou retrait de requêtes) sur le graphe de notre solution au cours de la phase d'exploration. On considère le graphe $G_t = (V, A)$ avec $|V|$ le nombre de sommets et $|S|$ le nombre moyen de successeurs directs. La méthode $\mathrm{topologicalSort}$ est un tri topologique du graphe en utilisant un parcours en profondeur dont la complexité est $\Theta(|V| + |A|)$. Le calcul des *slack time* (méthode $\mathrm{computeSlackTime}$), nécessitant le calcul de tous les plus courts chemins entre toutes les paires, est inspiré de l'algorithme de Floyd-Warshall [26] et a une complexité dont la borne supérieure est $\Theta(|V|^3)$ et est en moyenne $\Theta(|V| \times |S|^2)$. Le calcul des *forward time slack* (méthode $\mathrm{computeForwardTimeSlack}$), présenté dans l'équation 7.15, est effectué en $\Theta(|V| \times |S|)$. La mise à jour de la matrice de précédence (méthode $\mathrm{computePrecedenceMatrix}$) est effectuée lors de l'algorithme de Floyd-Warshall.

Algorithme 10 : FONCTION DE MISE À JOUR DU GRAPHE AVEC UN ORDONNANCEMENT AU PLUS TÔT : $\mathrm{updateEarliest}$

Entrée : un graphe de solution G_t

```
1 début
    /* Tri topologique de G_t                                   */
2   topologicalSort(G_t)
    /* Calcul des heures au plus tôt de G_t                     */
3   computeEarliestSchedule(G_t)
    /* Calcul des slack time                                    */
4   computeSlackTime(G_t)
    /* Calcul des forward time slack                            */
5   computeForwardTimeSlack(G_t)
    /* Calcul de la matrice de précédence                       */
6   computePrecedenceMatrix(G_t)
7 fin
```

L'algorithme de calcul du graphe G_t après sa modification a une complexité en moyenne de $\Theta(|V| \times |S|^2)$.

7.5 Expérimentations numériques

7.5.1 Instances

Cette partie présente la description des instances de tests utilisées pour la validation de l'algorithme. L'ALNS a été testé sur deux jeux d'instances différents. Le premier jeu vient des instances de Hirsch [57].

Le second jeu correspond à un jeu d'instances réelles de l'entreprise Luc Durand.

Instances de la littérature

Dans un premier temps, notre algorithme est testé sur les instances de Hirsch [57]. Le problème étudié, dans ce papier, est le Log Truck Scheduling Problem (LTSP) qui traite de la collecte et livraison en camions pleins avec fenêtres de temps. Ce problème n'intègre pas la synchronisation sur les ressources comme d'autres problèmes similaires de la littérature (El Hachemi et al. [44], Bredstrom et al. [18]). La flotte de camions est hétérogène sans coût fixe d'utilisation. 40 instances sont proposées et sont divisées en deux jeux. Le premier jeu contient 20 instances, numérotées de 1 à 20 (H1 - H20) et le deuxième jeu contient 20 instances, numérotées de 21 à 40 (H21 - H40).

Dans les deux jeux, 30 requêtes doivent être satisfaites avec l'utilisation de deux catégories différentes de camions. Des contraintes de compatibilités existent entre les camions et les requêtes dues à la limite de capacité du réseau routier. Ces instances représentent une journée de travail avec un horizon de temps défini dans l'intervalle $[0, 1440]$. Le réseau est composé de 589 sites. Pour chaque instance, il y a un site de collecte par requête et les livraisons sont regroupées sur uniquement 3 sites pour le jeu 1 et 4 sites pour le jeu 2. Dans ce dernier jeu d'instances, les requêtes ont des temps de service supérieurs et les camions ont des amplitudes de travail journalier supérieures (780 minutes contre 500 minutes).

Instances réelles

Nous testons également notre algorithme sur un jeu de données réelles issu de l'entreprise. 7 instances sont considérées et divisées en deux jeux. Les instances comportent 20 sites incluant un dépôt pour les véhicules. 3 catégories de camions sont disponibles pour servir les demandes (voir tableau 7.3).

Type	Nombre	Capacité	$\mathcal{CD}$	$\mathcal{CH}$	$\mathcal{CK}$
-	-	t	€	€/h	€/km
SEMI38	18	25	146.51	18.41	0.696
8X4	9	16	141.61	18.41	0.654
6X4	13	12	141.61	18.41	0.654

Tableau 7.3 – Flotte de camions des instances réelles.

Les instances OS22, OS32, OS49 contiennent à la fois des demandes à flux tendus et des demandes à flux détendus tandis que les instances OU35, OU51_1, OU51_2, OU85 contiennent uniquement des instances à flux détendus. Le tableau 7.4 présente les principales caractéristiques des instances. L'horizon de temps considéré est journalier de 6h à 20h ce qui correspond à une amplitude journalière de 14 heures. Nous considérons que les sites et le dépôt sont ouverts sans interruption. Le temps de service pour la collecte et la livraison est choisi à 10 minutes. L'instance OU85 correspond à un taux d'activité conséquent de l'entreprise.

7.5.2 Expérimentations

Dans cette partie, nous présentons les résultats numériques de notre algorithme ALNS. Nous détaillons, dans un premier temps l'initialisation de l'algorithme et le paramétrage de l'ALNS. Pour évaluer sa performance, nous testons l'algorithme sur des instances de la littérature, qui ont fait l'objet d'adaptations pour prendre en compte la synchronisation aux ressources. Pour chaque test et sur chaque instance, nous effectuons 10 exécutions ce qui nous permet d'obtenir la meilleure valeur et l'écart moyen des solutions à la meilleure valeur (obtenue par le solveur MIP ou bien l'algorithme ALNS). Dans un second temps, nous présentons les tests sur des instances réelles. Les tests ont été effectués sur une machine avec un processeur Xeon X5650 à 2.67 GHz et possédant 64 GB de RAM.

Instance	$\|R_s\|$	$\|R_u\|$	$\|R\|$
OS22	4	18	22
OS32	9	23	32
OS49	11	38	49
OU35	0	35	35
OU51_1	0	51	51
OU51_2	0	51	51
OU85	0	85	85

Tableau 7.4 – Principales caractéristiques des instances réelles. $|R_s|$ (respectivement $|R_u|$) représente le nombre de requêtes à flux tendus (resp. requêtes à flux détendus). $|R|$ représente le nombre total de requêtes.

Initialisation de l'ALNS

Pour construire une solution initiale pour l'ALNS, nous utilisons une des combinaisons possibles d'opérateurs de réparation. Nous testons les différentes combinaisons entre RequestInsertion, ResourceScheduling et RequestSelection. Comme, nous nous intéressons à la production de solutions pour l'entreprise, nous avons décidé de considérer uniquement les instances réelles.

Le tableau 7.5 montre le coût de la solution initiale pour les différentes combinaisons des critères RequestSelection et ResourceScheduling avec la méthode d'insertion sur les tournées **CI** et le tableau 7.6 avec la méthode d'insertion sur les tournées **ETA**. Les tableaux présente la déviation à la meilleure solution obtenue lors de ces tests.

RequestInsertion	**CI**													
ResourceScheduling	**EF**							**LF**						
RequestSelection	**EDD**	**EDRD**	**NSR**	**CR**	**CI**	**2R**	**3R**	**EDD**	**EDRD**	**NSR**	**CR**	**CI**	**2R**	**3R**
OS22	1.83	**0**	9.83	11.84	11.7	1.04	2.31	1.96	0.14	9.65	5.74	3.19	3.22	2.54
OS32	0.09	0.4	14.49	22.65	**0**	0.47	0.59	0.13	3.06	11.12	10.98	4.86	1.7	0.63
OS49	0.85	**0**	8.8	15.26	7.81	4.4	6.27	0.06	3.75	16.24	17.62	7.94	7.83	4.08
OU35	13.69	17.71	17.71	17.71	6.89	1.62	11	14.52	18.24	18.24	18.24	5.42	**0**	18.24
OU51_1	12	12	8.6	11.38	**0**	9.35	12.99	13.7	13.7	13.12	17.87	1.96	11.74	12.81
OU51_2	7.19	7.19	5.32	9	0.11	1.67	7.19	7.04	7.04	4.77	9.36	**0**	2	4.8
OU85	5.36	5.48	7.56	24.69	1.79	5.78	14.78	8.76	8.76	8.35	17.86	7.32	**0**	5.87
moy.	5.86	6.11	10.33	16.08	4.04	**3.48**	7.88	6.6	7.81	11.64	13.95	4.38	3.78	6.99

Tableau 7.5 – Résultat de la solution initiale des instances réelles avec l'opérateur d'insertion **CI**. Le pourcentage représente la déviation à la meilleure solution obtenue lors de ces tests uniquement.

RequestInsertion	**ETA**													
ResourceScheduling	**EF**							**LF**						
RequestSelection	**EDD**	**EDRD**	**NSR**	**CR**	**CI**	**2R**	**3R**	**EDD**	**EDRD**	**NSR**	**CR**	**CI**	**2R**	**3R**
OS22	141.42	141.42	123.79	132.17	124.06	132.61	132.74	141.42	141.42	123.79	132.17	124.06	132.61	132.74
OS32	78.97	78.97	82.8	63.17	73.98	71.07	71.07	78.97	78.97	82.8	63.17	73.98	71.07	71.07
OS49	56.8	56.8	31.97	35.75	45.97	50.59	50.59	56.8	56.8	31.97	35.75	45.97	50.81	50.81
OU35	51.7	51.7	47.98	47.98	48.53	47.98	47.98	52.23	52.23	48.52	48.52	50.2	48.52	48.52
OU51_1	19.07	19.07	40.52	18.74	26.1	19.07	19.07	19.07	19.07	40.52	18.74	20.18	19.07	19.07
OU51_2	40.82	40.82	45.77	26.34	28.12	27.52	39.47	40.82	40.82	45.77	25.51	28.12	28.87	39.47
OU85	30.93	30.93	33.86	28.29	30.53	31.65	29.34	30.59	30.59	33.86	26.51	31.95	31.65	29.27
moy.	59.96	59.96	58.1	50.35	53.9	54.35	55.75	59.99	59.99	58.17	50.05	53.49	54.66	55.85

Tableau 7.6 – Résultat de la solution initiale des instances réelles avec l'opérateur d'insertion **ETA**. Le pourcentage représente la déviation à la meilleure solution obtenue lors de ces tests uniquement.

En s'appuyant sur ces tests, nous choisissons d'utiliser l'opérateur de réparation **2R-CI-EF** : les requêtes sont triées en utilisant la règle **2R**, l'insertion dans la tournée est réalisée grâce à l'opérateur de meilleure

insertion **CI** et l'ordonnancement aux ressources est effectué au plus tôt **EF**.

L'opérateur **ETA** n'est pas un bon candidat pour la création d'une solution initiale. Cet opérateur tend à créer des tournées, induisant un coût supplémentaire dans la fonction objectif, par rapport à positionner les requêtes à leurs meilleures emplacements.

Choix des opérateurs de réparation

L'ensemble des opérateurs de réparation ($\mathcal{N}^+$) comprend des combinaisons des trois règles de décisions décrites dans la section 7.2.6. Certaines combinaisons se trouvent être contre intuitive pour l'ordonnancement sur les ressources. En effet, il est incohérent d'insérer une requête à la fin d'une tournée (**ETA**) et ces points de collecte et livraison au début de leur ressource (**EF**). Le critère **CI** ne présente pas de bonnes solutions initiales (voir tableau 7.5) lorsqu'il est combiné avec le critère **LF**. Néanmoins, nous souhaitons les conserver et les associés avec les critères d'insertion sur les tournées **CI** et **2R**. La combinaison **CI+LF** va diversifier le parcours des solutions en proposant un ordonnancement différent sur les ressources, tout comme la combinaison **ETA+LF** avec des solutions ayant tendance à avoir comporter plus de véhicules.

Par rapport aux travaux de la littérature, qui majoritairement, utilisent des insertions basés sur la meilleure insertion, nous faisons le choix d'avoir plus d'opérateurs de réparation pour diversifier la recherche.

Au final, nous considérons les douze combinaisons suivantes comme opérateurs de réparation :

RequestInsertion		**CI**		**ETA**
ResourceScheduling		**EF**	**LF**	**LF**
RequestSelection	**EDD**	✓		
	EDRD	✓		
	NSR	✓		
	CI	✓	✓	✓
	2R	✓	✓	✓
	3R	✓		
	CR	✓		✓

Tableau 7.7 – Combinaisons utilisées pour les opérateurs de réparation.

Calibrage de l'ALNS

Un parcours de la littérature sur l'ALNS montre différentes façons de calibrer cet algorithme en fonction du type de problème résolu. Beaucoup de paramètres peuvent être ajustés pour fournir une meilleure solution. Dans cette optique, nous choisissons de nous concentrer sur quatre ensembles de paramètres :

1. la couche adaptative ;
2. le bruitage ;
3. la quantité de requêtes retirées dans la phase de destruction ;
4. la performance des opérateurs de destruction.

Tous ces tests de calibrage ont été effectués sur un sous-ensemble d'instances en considérant 10 exécutions de l'algorithme pour chaque instance. Le premier sous-ensemble contient cinq instances de Hirsch [57] (H6 - H10 - H26 - H29 - H34) et le second sous-ensemble contient trois instances industrielles comportant des flux continus (OS22 - OS32 - OS49). Le critère d'arrêt utilisé de notre ALNS est la limite de temps fixée à 10 minutes $t_{limit} = 600s$ ce qui représente un bon compromis vis à vis du temps alloué à l'algorithme dans un contexte d'utilisation industrielle. De plus, ce critère permet une comparaison équitable des opérateurs, indépendamment de leur complexité.

Pour chaque test de calibrage, nous comparons les solutions obtenues avec la meilleure solution obtenue. Celle-ci étant définie comme la meilleure solution trouvée parmi tous les résultats obtenus lors du test de calibrage considéré. Lorsque nous choisissons un paramètre à l'issue d'un test, nous le conservons pour les tests futurs.

La couche adaptative de l'ALNS La probabilité initiale de sélection des opérateurs est fixée à $\frac{1}{|\mathcal{N}^-|}$ pour les opérateurs de destruction et à $\frac{1}{|\mathcal{N}^+|}$ pour les opérateurs de réparation. La probabilité est mise à jour en suivant la méthode utilisée dans [87]. À chaque itération de l'ALNS, le score de l'opérateur i sélectionné est augmenté de la quantité, au choix, $\sigma_1, \sigma_2, \sigma_3$ suivant la procédure d'adaptation des scores présentée dans [87]. La recherche est divisée en segments temporels de 100 itérations chacun. À chaque fin d'un segment, la probabilité de sélection des opérateurs est mise à jour et leurs scores respectifs sont réinitialisés à 0.

En se basant sur différents papiers de la littérature, nous avons testé quatre configurations de la couche adaptative, présentées dans le tableau 7.8. La première configuration, appelée "AD0", n'implémente pas la couche adaptative et les scores des opérateurs restent constants. Ce choix s'est avéré pertinent dans les travaux de Léhuédé et al. [64]. La seconde configuration "AD1" est obtenue à partir du travail de Demir et al. [30]. La troisième configuration "AD2" est la configuration originale proposée par Ropke et Pisinger in [87]. Enfin, la dernière configuration"AD3" est obtenue à partir des travaux de Masson et al. [72].

Configuration	σ_1	σ_2	σ_3
$AD0$	-	-	-
$AD1$	1	0	5
$AD2$	33	9	13
$AD3$	33	20	13

Tableau 7.8 – Description des quatre configurations de la couche adaptative testées avec un temps limite $t_{limit} = 600s$.

Les tests ont été exécutés avec les paramètres listés dans le tableau 7.9.

Paramètre	Valeur
Solution initiale	**2R-CI-EF**
κ	100000
t_{limit}	$600s$
bruitage	non
ξ_{min}	0.10
ξ_{max}	0.40
c	0.99975

Tableau 7.9 – Présentation des paramètres retenus pour l'exécution des tests sur la couche adaptative.

Le tableau 7.10 compare les quatre configurations sur les instances de tests, en précisant le pourcentage de déviation par rapport à la meilleure solution connue *GAP*.

Les résultats confirment une légère influence de la couche adaptative sur l'efficacité de l'ALNS (l'écart entre les moyennes est de l'ordre de 0.3%). La configuration"AD3" présente les meilleurs résultats comparée aux autres configurations. Cette configuration propose la meilleure déviation moyenne et est la meilleure solution quatre fois sur huit. Nous choisissons d'utiliser la configuration "AD3" pour les expérimentations numériques.

Des travaux récents [52] ont montré que la couche adaptative de l'ALNS avait une contribution mineure sur les performances de l'algorithme. Néanmoins, nous utilisons un nombre important d'opérateurs et, dans

Instance	GAP_{AD0} (%)	GAP_{AD1} (%)	GAP_{AD2} (%)	GAP_{AD3} (%)
OS22	**0.25**	0.90	0.77	0.65
OS32	**0.01**	**0.01**	0.06	0.17
OS49	0.98	1.44	**0.67**	0.68
H6	0.70	1.06	1.07	**0.64**
H10	0.73	1.09	0.74	**0.63**
H26	2.22	2.69	2.44	**1.94**
H29	1.21	1.19	**0.92**	1.10
H34	1.26	1.08	0.97	**0.85**
moy.	0.92	1.18	0.96	**0.83**

Tableau 7.10 – Résultats pour quatre configurations de la couche adaptative : le tableau présente le pourcentage moyen de déviation à la meilleure solution connue avec $t_{limit} = 600s$ pour chaque configuration sur les huit instances de test.

notre cas, la couche adaptative permet de réduire la contribution de certains opérateurs qui seraient moins efficaces que d'autres.

Influence du bruit Nous proposons d'étudier l'influence du bruit dans les opérateurs de réparation. Le coût d'insertion d'une requête peut être calculé sans le bruit (C) ou avec le bruit (C'). Nous nous intéressons à la déviation moyenne en pourcentage de la fonction objectif à la meilleure solution pour chaque instance.

Les tests ont été exécutés avec les paramètres listés dans le tableau 7.11.

Paramètre	Valeur
Solution initiale	**2R-CI-EF**
κ	100000
t_{limit}	$600s$
σ_1	33
σ_2	20
σ_3	13
ξ_{min}	0.10
ξ_{max}	0.40
c	0.99975

Tableau 7.11 – Présentation des paramètres retenus pour l'exécution des tests sur l'influence du bruit.

Le tableau 7.12 montre que la contribution du bruit n'est pas significativement améliorante. Des résultats similaires ont été observés par Bortfeldt et al. dans [16], où le bruit a un impact mineur sur la qualité de la solution, et décident de ne pas l'utiliser.

Malgré des tests mitigés où la différence n'est pas grande, nous choisissons d'utiliser le bruit dans la suite de nos tests.

Nombre de requêtes retirées L'influence de la quantité de requêtes retirées q à chaque itération a été étudiée par Demir et al. [30] et par Masson et al. [72]. Nous proposons cinq configurations différentes, en sélectionnant le pourcentage de requêtes à retirer dans l'intervalle $[\xi_{min}, \xi_{max}]$, avec ξ_{min} la borne inférieure et ξ_{max} la borne supérieure en pourcentage du nombre de requêtes à retirer. Le tableau 7.14 présente le pourcentage moyen de déviation à la meilleure solution connue pour cinq configurations sur les instances de calibrage.

Les tests ont été exécutés avec les paramètres listés dans le tableau 7.13.

Instance	$GAP_{without}$ (%)	GAP_{with} (%)
OS22	1.04	**0.65**
OS32	**0.13**	0.23
OS49	0.38	0.38
H6	**0.68**	0.92
H10	**0.53**	1.03
H26	2.83	**2.43**
H29	**0.48**	0.54
H34	0.82	**0.70**
moy.	0.86	0.86

Tableau 7.12 – Résultats de l'influence du bruit (avec et sans) en comparant la déviation moyenne à la meilleure solution connue avec $t_{limit} = 600s$.

Paramètre	Valeur
Solution initiale	**2R-CI-EF**
κ	100000
t_{limit}	$600s$
bruitage	oui
σ_1	33
σ_2	20
σ_3	13
c	0.99975

Tableau 7.13 – Présentation des paramètres retenus pour l'exécution des tests sur le nombre de requêtes retirées.

Instance	$[5, 10]$	$[10, 20]$	$[10, 35]$	$[10, 40]$	$[10, 50]$
OS22	0.82	0.51	0.82	0.79	**0.43**
OS32	0.14	0.06	**0.01**	0.17	0.34
OS49	0.74	**0.42**	0.53	0.58	0.56
H6	**0.64**	0.66	0.73	0.75	0.73
H10	1.10	1.03	0.97	**0.68**	0.73
H26	3.14	**2.36**	2.68	2.55	2.72
H29	0.99	0.97	1.10	1.34	**0.92**
H34	1.38	1.15	1.15	**0.90**	1.22
moy.	1.12	**0.89**	1.00	0.97	0.96

Tableau 7.14 – Résultats pour cinq configurations du nombre de requêtes retirées : le tableau présente le pourcentage moyen de déviation à la meilleure solution connue avec $t_{limit} = 600s$.

Le tableau 7.14 nous montre que la sensibilité de l'ALNS vis à vis de ce paramètre fait varier le pourcentage moyen de déviation de $\pm 0.2\%$. Un intervalle est meilleur que les autres en moyenne, celui de $[10, 20]$. Nous choisissons la quantité de requêtes à enlever entre 10% et 20% par rapport au nombre total de requêtes.

Cette configuration a l'avantage de permettre plus d'itérations de l'ALNS car les voisinages d'exploration sont plus petits en moyenne.

Performance des opérateurs de destruction Enfin, nous étudions la performance des opérateurs de destruction sur l'ALNS. Pour chaque test, nous exécutons l'ALNS avec tous les opérateurs de destruction sauf un. Le tableau 7.16 montre le pourcentage moyen de déviation à la meilleure solution connue pour chaque instance et chaque configuration.

Les tests ont été exécutés avec les paramètres listés dans le tableau 7.15.

Paramètre	Valeur
Solution initiale	**2R-CI-EF**
κ	100000
t_{limit}	$600s$
bruitage	oui
σ_1	33
σ_2	20
σ_3	13
ξ_{min}	0.10
ξ_{max}	0.20
c	0.99975

Tableau 7.15 – Présentation des paramètres retenus pour l'exécution des tests sur la performance des opérateurs de destructions.

Instance	**RDO**	**WDO**	**DRDO**	**TRDO**	**CDO**	**RODO**	**REDO**	**ALL**
OS22	**0.08**	0.74	0.82	0.82	0.42	0.90	1.35	0.43
OS32	0.14	0.04	**0.02**	0.40	0.24	0.06	0.27	0.20
OS49	0.44	0.73	0.40	0.47	0.38	**0.33**	0.35	0.42
H6	0.83	0.75	0.55	0.73	0.80	**0.51**	0.52	0.71
H10	0.86	0.81	0.91	0.92	0.94	**0.59**	0.82	0.75
H26	3.51	3.29	3.76	3.45	**2.71**	3.14	2.82	3.46
H29	**0.96**	1.19	1.07	1.44	1.19	1.24	1.22	1.04
H34	1.13	1.26	1.11	**0.86**	1.08	1.29	1.10	1.19
moy.	0.99	1.10	1.08	1.14	**0.97**	1.01	1.06	1.02

Tableau 7.16 – Influence des opérateurs de destruction sur la déviation moyenne à la meilleure solution connue. Chaque colonne indique la performance de l'ALNS sans l'opérateur de destruction indiqué en en-tête avec $t_{limit} = 600s$.

Les résultats ne montrent pas beaucoup d'écart entre les différentes configurations (variabilité de 0.1% par rapport à la configuration avec tous les opérateurs). Nous pouvons noter les meilleurs résultats obtenus sans l'opérateur de destruction **CDO**. Néanmoins, cet opérateur peut apporter une efficacité dans la destruction des solutions. Nous choisissons de conserver tous les opérateurs dans les expérimentations en se basant sur le fait que cette configuration présente les meilleurs résultats de calibrage en moyenne.

Paramètres finaux de ALNS

Les solutions proposées par la phase de calibrage montrent une certaine variabilité de l'algorithme. L'ALNS n'est donc pas très sensible aux différentes valeurs des paramètres.

Pour conclure ce calibrage des paramètres sur les instances de test, nous choisissons les paramètres de la roulette biaisée ainsi $(\sigma_1, \sigma_2, \sigma_3, r) = (33, 20, 13, 0.1)$ où r est le facteur de réaction qui contrôle l'ajustement du poids [87]. Le mécanisme de roulette biaisée est réinitialisé toutes les 100 itérations. Comme critères d'arrêt, le nombre total d'itérations de l'algorithme est $\Phi = 25000$, le nombre total d'itérations sans amélioration est $\tau = 5000$ et le temps total de l'algorithme est de $t_{limit} = 3600s$. Le recuit simulé commence avec une température initiale qui est calculée en fonction de la valeur de la fonction objectif de la solution initiale. À chaque itération, le pourcentage de requêtes retirées est choisi dans l'intervalle $(\xi_{min}, \xi_{max}) = (0.1, 0.2)$. Enfin, le bruit est ajouté dans les opérateurs de réparation.

Comparaison avec les instances de la littérature

Pour évaluer les performances de notre algorithme ALNS, nous le comparons avec l'algorithme de recherche tabou de Hirsch [57] sur les instances de Hirsch pour le LTSP. Nous présentons les résultats de cette comparaison dans les tableaux 7.17 et 7.18. La méthode de recherche tabou est nommée TS-HIRSCH. Notre algorithme ALNS sans les contraintes de synchronisation est noté $ALNS_0$.

Les comparaisons sont faites sur la base de la meilleure solution obtenue pour dix exécutions de l'ALNS sur chaque instance. La colonne "LB" fournit une borne inférieure pour chaque instance, borne calculée par [57] avec une relaxation du problème résolue par le solveur Xpress-MP. Les colonnes "Best value" indiquent la valeur obtenue par la méthode associée. GAP_{LB} indique la déviation en pourcentage de la meilleure solution obtenue par la méthode par rapport à la borne inférieure. GAP_H indique la déviation en pourcentage de $ALNS_0$ sur TS-HIRSCH et est défini par $GAP_H = \frac{Best_{ALNS_0} - Best_{TS-HIRSCH}}{Best_{TS-HIRSCH}}$. Une valeur positive indique une moins bonne "Best value" de notre méthode et une valeur négative indique une meilleure solution.

Comme le montrent les tableaux 7.17- 7.18, notre algorithme ALNS est compétitif par rapport au TS-HIRSCH. L'ALNS est capable de trouver quatre nouvelles meilleures solutions dans le jeu 1 et dix nouvelles meilleures solutions dans le jeu 2. Même si l'algorithme n'améliore pas toutes les solutions, la déviation moyenne est minime avec 0.00% pour le jeu 1 (l'ALNS est équivalent au TS-HIRSCH pour 14 instances) et -0.14% pour le jeu 2. Le temps de calcul moyen par instance est de 1500s en moyenne. Ce temps est inférieur au critère d'arrêt $t_{limit} = 3600s$, l'algorithme s'est arrêté car un des autres paramètres a été atteint (nombre d'itérations maximum Φ ou nombre d'itérations sans amélioration τ). Notre algorithme ALNS a été développé pour un problème plus compliqué que celui proposé dans [57].

Ces tableaux confirment l'efficacité de notre algorithme ALNS.

Impact des contraintes de synchronisation aux ressources

Pour évaluer l'influence des contraintes de synchronisation aux ressources, nous avons modifié le problème LTSP, en considérant tous les sites comme des ressources de capacité unaire : un seul camion peut charger ou décharger un matériau à chaque instant sur une ressource donnée. Le temps d'occupation sur la ressource est alors égal au temps de service. Les instances correspondantes sont suffixées par "-RS". Pour prendre en charge les contraintes de synchronisation, nous ajoutons les opérateurs spécifiques qui sont liés aux ressources dans notre ALNS, à savoir l'opérateur de destruction **REDO** et les opérateurs de réparation **CR+CI+EF / CR+ETA+LF**. Cette version modifiée de l'ALNS avec ces opérateurs de synchronisation est notée $ALNS_1$.

Nous comparons les résultats de ces tests sur les instances "-RS" (jeu 1 et jeu 2) avec les meilleurs résultats obtenus avec $ALNS_0$.

Les tableaux 7.19 et 7.20 présentent la comparaison entre la meilleure solution connue pour les instances sans les contraintes de synchronisation aux ressources ($ALNS_0$), colonne "BKS", et notre algorithme, co-

		TS-HIRSCH		$ALNS_0$		
Instance	LB	Best value	$GAP_{LB}\%$	Best value	$GAP_{LB}\%$	$GAP_H\%$
H1	2593.37	2596.56	0.12	**2596.13**	0.11	**-0.02**
H2	2371.83	2380.41	0.36	**2378.64**	0.29	**-0.07**
H3	1969.26	**1998.42**	1.48	**1998.42**	1.48	**0**
H4	2399.06	2405.79	0.28	**2405.43**	0.27	**-0.01**
H5	2191.48	**2217.85**	1.20	**2217.85**	1.20	**0**
H6	2496.35	**2610.76**	4.58	**2610.76**	4.58	**0**
H7	2430.39	**2529.03**	4.06	**2529.03**	4.06	**0**
H8	2225.52	**2251.39**	1.16	**2251.39**	1.16	**0**
H9	2298.11	**2324.46**	1.15	**2324.46**	1.15	**0**
H10	2394.29	**2489.35**	3.97	2492.82	4.12	0.14
H11	2264.16	2290.32	1.16	**2290.03**	1.14	**-0.01**
H12	2529.86	**2542.42**	0.50	**2542.42**	0.50	**0**
H13	2305.70	**2309.81**	0.18	**2309.81**	0.18	**0**
H14	2264.43	**2320.79**	2.49	**2320.79**	2.49	**0**
H15	2225.52	**2325.69**	4.5	**2325.69**	4.50	**0**
H16	2274.25	**2367.86**	4.12	**2367.86**	4.12	**0**
H17	2382.47	**2481.33**	4.15	**2481.33**	4.15	**0**
H18	2280.96	**2325.52**	1.95	**2325.52**	1.95	**0**
H19	2656.41	**2736.10**	3.00	**2736.10**	3.00	**0**
H20	2227.70	**2233.71**	0.27	2234.69	0.31	0.04
moy.	-	-	2.03	-	2.04	0.00

Tableau 7.17 – Résultats obtenus par la méthode ALNS comparés aux résultats obtenues par [57] pour les instances du jeu 1.

	TS-HIRSCH	$ALNS_0$	
Instance	Best value	Best value	$GAP_H\%$
H21	4749.97	**4747.32**	**-0.06**
H22	**5425.99**	**5425.99**	**0**
H23	**4866.23**	**4866.23**	**0**
H24	**5255.53**	**5255.53**	**0**
H25	6103.50	**6101.86**	**-0.03**
H26	4686.41	**4652.96**	**-0.71**
H27	**6058.37**	**6058.14**	**0**
H28	**5151.54**	**5151.54**	**0**
H29	**5845.09**	**5845.09**	**0**
H30	5347.46	**5340.69**	**-0.13**
H31	5059.85	**5058.21**	**-0.03**
H32	**4691.05**	**4691.05**	**0**
H33	4919.39	**4911.39**	**-0.16**
H34	**5529.69**	**5529.69**	**0**
H35	6469.63	**6466.60**	**-0.05**
H36	4878.02	**4872.66**	**-0.11**
H37	5595.49	**5566.61**	**-0.52**
H38	**5648.81**	**5649.04**	**0**
H39	**5111.85**	**5111.85**	**0**
H40	4677.23	**4626.24**	**-1.09**
moy.	-	-	-0.14

Tableau 7.18 – Résultats obtenus par la méthode $ALNS_0$ comparés aux résultats obtenues par [57] pour les instances du jeu 2.

lonne "$ALNS_1$". Pour ces tests, nous considérons la meilleure solution obtenue parmi dix exécutions de notre algorithme sur chaque instance avec les mêmes critères d'arrêt ($\Phi = 25000$, $\tau = 5000$ et $t_{limit} = 3600s$).

	BKS	$ALNS_1$	
Instance	Best value	Best value	$GAP_{BKS}\%$
H1-RS	2596.13	2599.48	0.13
H2-RS	2378.64	2381.27	0.11
H3-RS	1998.42	2010.33	0.60
H4-RS	2405.43	2408.18	0.11
H5-RS	2217.85	2226.63	0.40
H6-RS	2610.76	2625.30	0.56
H7-RS	2529.03	2535.11	0.24
H8-RS	2251.39	2258.50	0.32
H9-RS	2324.46	2325.45	0.04
H10-RS	2489.35	2503.56	0.57
H11-RS	2290.03	2296.09	0.26
H12-RS	2542.42	2547.34	0.19
H13-RS	2309.81	2312.44	0.11
H14-RS	2320.79	2324.37	0.15
H15-RS	2325.69	2329.19	0.15
H16-RS	2367.86	2371.86	0.17
H17-RS	2481.33	2486.66	0.21
H18-RS	2325.52	2335.41	0.43
H19-RS	2736.10	2747.42	0.41
H20-RS	2233.71	2238.50	0.21
moy.	-	-	0.27

Tableau 7.19 – Impact des contraintes de synchronisation aux ressources. Résultats obtenus par $ALNS_1$ (avec synchronisation) comparés avec la meilleure solution obtenue sur les instances du jeu 1.

Les résultats montrent que les contraintes de synchronisation aux ressources ont un impact mineur sur la valeur de la fonction objectif sur ces instances. En particulier, la déviation moyenne aux meilleures solutions connues avec la synchronisation est de 0.27% pour les instances H1-RS - H20-RS et 0.42% pour les instances H21-RS - H40-RS. Nous observons que les temps de calcul diminuent fortement avec une moyenne à 500 secondes. Nous expliquons cette diminution du temps de calcul par l'absence de découverte de meilleure solution par notre algorithme. Les contraintes de synchronisation complexifie le problème et des solutions équivalente sans être meilleures doivent être visitées. Pour aller plus loin dans l'évaluation de l'impact des ressources de synchronisation sur les instances de Hirsch, nous avons multiplié le temps de service des points de livraison par un facteur appelé *resource occupancy factor* $f \in \{0.5, 1, 1.5, 2, 2.5, 3\}$. Nous modifions uniquement le temps de service sur les livraisons puisque, dans ce problème, seuls les sites de livraisons sont communs à plusieurs requêtes. Ces sites correspondent à des usines où les camions de transport de bois déchargent. Dix exécutions de l'algorithme sont effectuées pour chaque instance avec les mêmes critères d'arrêt ($\Phi = 25000$, $\tau = 5000$ et $t_{limit} = 3600s$).

Une figure synthétique de l'écart moyen par rapport à la meilleure solution est présentée dans la figure 7.3. Nous observons, qu'en moyenne, l'écart à la meilleure solution connue sans la synchronisation est correlé à l'augmentation de l'occupation des ressources. Cependant, cet écart n'augmente pas de plus de 2%, même quand le facteur d'occupation est multiplié par 3.

Cette conclusion doit être faite conjointement avec le nombre de véhicules utilisés dans chaque solution. Dans les instances de Hirsch, un maximum de dix camions est autorisé pour chaque instance et pour chaque camion un temps maximal de service est donné. Aussi, aucun coût fixe journalier n'est donné. La

	BKS	$ALNS_1$	
Instance	Best value	Best value	$GAP_{BKS}\%$
H21-RS	4747.32	4760.41	0.28
H22-RS	5425.99	5426.54	0.01
H23-RS	4866.23	4866.23	0.00
H24-RS	5255.53	5280.33	0.47
H25-RS	6101.86	6124.25	0.37
H26-RS	4652.96	4725.18	1.55
H27-RS	6058.14	6094.37	0.60
H28-RS	5151.54	5172.73	0.41
H29-RS	5845.09	5865.07	0.34
H30-RS	5340.69	5352.39	0.22
H31-RS	5058.21	5060.08	0.04
H32-RS	4691.05	4692.46	0.03
H33-RS	4911.39	4936.19	0.50
H34-RS	5529.69	5541.32	0.21
H35-RS	6466.60	6511.03	0.69
H36-RS	4872.66	4890.20	0.36
H37-RS	5566.61	5603.37	0.66
H38-RS	5648.81	5673.83	0.44
H39-RS	5111.85	5111.85	0.00
H40-RS	4626.24	4684.43	1.26
moy.	-	-	0.42

Tableau 7.20 – Impact des contraintes de synchronisation aux ressources. Résultats obtenus par $ALNS_1$ (avec synchronisation) comparés avec la meilleure solution obtenue sur les instances du jeu 2.

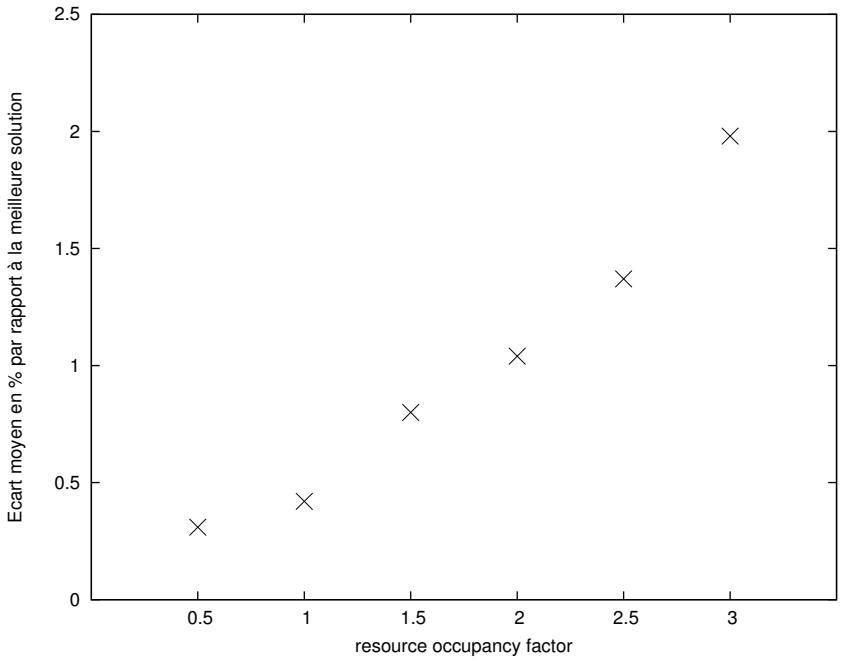


FIGURE 7.3 – Influence du *resource occupancy factor* sur l'écart moyen en pourcentage à la meilleure solution sur les instances H1-RS - H40-RS.

figure 7.4 montre le nombre moyen de camions utilisés dans chaque solution de chaque instance par rapport au facteur f. Nous observons que l'augmentation du temps d'occupation sur les ressources a un impact non négligeable. La taille limitée de la flotte empêche la réalisabilité de la solution lorsque les temps de service augmentent. De $f = 2.5$, une requête n'est pas servie à la fin de l'algorithme ALSN pour l'instance H27-RS. C'est aussi le cas pour $f = 3$ pour l'instance H29-RS. Il faut alors des camions supplémentaires dans la flotte pour servir ces requêtes.

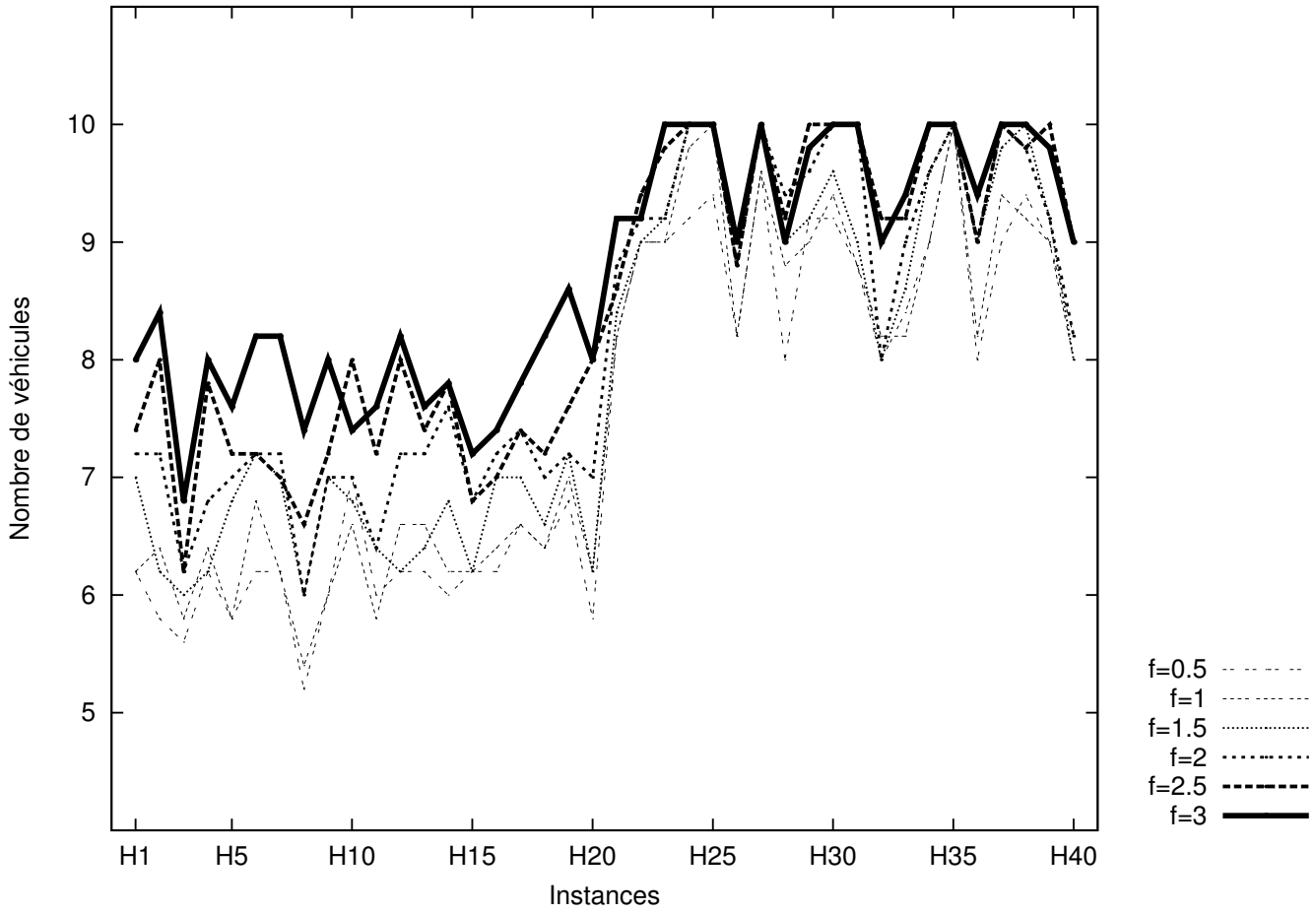


FIGURE 7.4 – Influence du *resource occupancy factor* sur le nombre de véhicules H1-RS - H40-RS.

Tests sur les instances réelles

Dans cette partie, nous proposons de tester notre algorithme ALNS avec les paramètres définis par la phase de calibrage sur les instances réelles de l'entreprise de travaux publics (instances appelées "O"). Deux séries de tests sont effectuées.

Résultats de la résolution exacte du problème par un solveur MIP Nous avons choisi de résoudre notre problème FT-PDP-RS, décrit dans la section 7.1, avec un solveur de type MIP (GUROBI 6.5.1) sur les instances réelles. Nous avons lancé deux exécutions :

1. une exécution de 10 minutes où nous sauvegardons la meilleure solution obtenue par le solveur (G_{600}) ;
2. une exécution de 2 heures où nous sauvegardons la meilleure solution obtenue par le solveur (G_{7200}).

Le tableau 7.21 présente les résultats obtenus par le solveur avec les deux exécutions ainsi qu'une borne inférieure proposée par le solveur (G_{LB}).

Instance	G_{LB}	600s		7200s	
		G_{600}	$GAP_{LB}\%$	G_{7200}	$GAP_{LB}\%$
OS22	2446.00	2776.95	13.53	2546.44	4.11
OS32	4527.28	4909.84	8.45	4820.26	6.47
OS49	6159.78	6992.72	13.52	6806.04	10.49
OU35	2368.73	3996.12	68.70	3545.65	49.69
OU51_1	3879.12	6507.23	67.75	5870.73	51.34
OU51_2	3596.37	6013.83	67.22	5892.94	63.86
OU85	3938.40	-	-	-	-

Tableau 7.21 – Résolution exacte des instances avec le solveur GUROBI. Le tableau présente la borne inférieure fournie par le solveur (G_{LB}), la meilleure solution obtenue au bout de 600s (G_{600}) et son écart en $GAP_{LB}\%$ à la borne inférieure (colonne 4), la meilleure solution obtenue au bout de 7200s (G_{7200}) et son écart à la borne inférieure en $GAP_{LB}\%$ (colonne 6).

Nous notons que le solveur GUROBI n'a pas trouvé de solution optimale dans le temps imparti (10 minutes ou 2 heures). Donner plus de temps au solveur (passer de 10min à 2h) permet de trouver de meilleures solutions. La borne inférieure fournie par le solveur est de mauvaise qualité sur les instances OU, ce qui est surprenant pour un modèle de tournées à 3 indices comportant plusieurs contraintes linéarisées. A contrario, la borne inférieure est relativement bonne sur les instances OS.

Résultats de l'ALNS avec les paramètres définis par la phase de calibrage La meilleure solution trouvée jusqu'à présent parmi les tests effectués (solveur et ALNS) est notée BKS.

Pour ces tests, nous utilisons les paramètres définis par la phase de calibrage. Le tableau 7.22 présente les résultats de l'ALNS sur ces instances. Pour chaque instance, dix exécutions ont été réalisées et nous retournons la meilleure solution trouvée par l'ALNS A_{25000}, la déviation en % à la meilleure solution entre les tests de l'ALNS et la résolution par le solveur (G_{7200}), le nombre de tournées dans la solution ($\#_{route}$) et le temps d'exécution de l'algorithme en secondes.

Instance	BKS	G_{7200}	$\#_{route}$	A_{25000}	GAP_{BKS} (%)	$\#_{route}$	*time* (s)
OS22	2546.44	**2546.44**	4	2779.92	9.17	5	60
OS32	4820.26	**4820.26**	9	4840.26	0.41	9	162
OS49	6384.11	6806.04	13	**6384.11**	0	11	655
OU35	3276.03	3545.65	8	**3276.03**	0	6	182
OU51_1	5374.81	5870.73	13	**5374.81**	0	10	615
OU51_2	4985.63	5892.94	14	**4985**.63	0	9	540
OU85	5902.89	0	0	**5902.89**	0	11	1506

Tableau 7.22 – Résultats obtenus par l'ALNS sur les instances réelles. Ce tableau présente la meilleure solution connue (BKS), la meilleure solution trouvée par le solveur (G_{7200}), la meilleure solution trouvée par l'ALNS A_{25000}, son écart à BKS en % et son temps d'exécution en s. Nous renseignons le nombre de tournées dans chaque solution dans la colonne ($\#_{route}$). Les critères d'arrêt sont ceux de la section 7.5.2 ($\Phi = 25000$, $\tau = 5000$ et $t_{limit} = 3600s$).

Ces résultats montrent que l'ALNS est performant en général puisqu'il trouve de meilleures solutions par rapport au solveur exact. Nous remarquons que les temps d'exécutions augmentent avec la taille des instances en nombre de requêtes. Toutefois, le solveur est relativement meilleur que notre algorithme ALNS sur l'instance de petite taille OS22 où il trouve la meilleure solution alors que notre algorithme est loin à 9% de cette meilleure solution. Mais cette différence est expliquée par le nombre de tournées dans la solution.

Pour cette instance OS22, le solveur trouve une solution à 4 tournées alors que notre algorithme ALNS trouve une solution à 5 tournées. La différence d'utilisation d'un véhicule (146.51€) constitue 63% de la différence de coût entre les deux solutions (233.48€).

Impact du nombre d'itérations Nous proposons dans cette partie de discuter la performance de l'ALNS avec $\Phi = 50000$ comme unique critère d'arrêt. Dix exécutions de chaque instance réelle ont été effectuées. La meilleure solution trouvée par l'ALNS est notée A_{50000}. Nous comparons les résultats obtenus avec ceux de la configuration du paragraphe précédent.

Le tableau 7.23 présente les résultats en comparant les meilleures solutions. Le tableau présente les résultats en comparant la moyenne des 10 exécutions.

				$\Phi = 25000$				$\Phi = 50000$			
Instance	BKS	G_{7200}	$\#_{route}$	A_{25000}	GAP_{BKS} (%)	$\#_{route}$	*time* (s)	A_{50000}	GAP_{BKS} (%)	$\#_{route}$	*time* (s)
OS22	2546.44	**2546.44**	4	2779.92	9.17	5	60	2779.92	9.17	5	556
OS32	4820.26	**4820.26**	9	4840.26	0.41	9	162	4839.84	0.41	9	1127
OS49	6364.38	6806.04	13	6384.11	0.31	11	655	**6364.38**	0	11	3825
OU35	3264.27	3545.65	8	3276.03	0.36	6	182	**3264.27**	0	6	1566
OU51_1	5374.81	5870.73	13	5374.81	0.00	10	615	**5374.81**	0	10	3665
OU51_2	4950.18	5892.94	14	4985.63	0.72	9	540	**4950.18**	0	9	4208
OU85	5725.13	-	-	5902.89	3.10	11	1506	**5725.13**	0	11	3601

Tableau 7.23 – Résultats obtenus par l'ALNS sur les instances réelles. Ce tableau présente la meilleure solution connue (BKS), la meilleure solution trouvée par le solveur (G_{7200}), la meilleure solution trouvée par l'ALNS A_{25000}, son écart à BKS en % et son temps d'exécution en s et la meilleure solution trouvée par l'ALNS A_{50000}, son écart à BKS en % et son temps d'exécution en s. Nous renseignons le nombre de tournées dans chaque solution dan la colonne ($\#_{route}$). Le critère d'arrêt est $\Phi = 50000$.

Les résultats montrent que la solution est améliorée dans tous les cas en comparaison avec le tableau 7.22. Les temps de calcul sont toutefois supérieurs puisque nous avons autorisé 50000 itérations. Dans un contexte d'utilisation industrielle de cette méthode, le gain obtenu ne vaut pas la peine d'augmenter le nombre d'itérations. Les résultats de A_{25000} restent très bons vis à vis de la meilleure solution BKS. Nous observons également que notre algorithme ne trouve pas la solution avec 4 tournées pour l'instance OS22.

Enfin, dans le tableau 7.24, nous comparons la valeur moyenne des solutions pour les dix exécutions par rapport à la meilleure des solutions trouvées dans chacune des configurations (ALNS avec $\Phi = 25000$, $\tau = 5000$, $t_{limit} = 3600s$ ou ALNS avec $\Phi = 50000$).

	$\Phi = 25000$			$\Phi = 50000$		
Instance	A_{25000}	*moy.*	$GAP_{\mathrm{A}_{25000}}$ (%)	A_{50000}	*moy.*	$GAP_{\mathrm{A}_{50000}}$ (%)
OS22	2779.92	2798.24	0.66	2779.92	2791.62	0.42
OS32	4840.26	4845.66	0.11	4839.84	4840.13	0.01
OS49	6384.11	6403.78	0.31	6364.38	6378.46	0.22
OU35	3276.03	3370.98	2.90	3264.27	3266.62	0.07
OU51_1	5374.81	5386.12	0.21	5374.81	5375.38	0.01
OU51_2	4985.63	5048.11	1.25	4950.18	4969.93	0.40
OU85	5902.89	6054.47	2.57	5725.13	5817.6	1.62
moy.			1.14			**0.39**

Tableau 7.24 – Ce tableau présente la meilleure solution, la moyenne des solutions et la déviation de la moyenne à la meilleure solution pour les deux configurations de l'ALNS $\Phi = 25000$, $\tau = 5000$, $t_{limit} = 3600s$ et $\Phi = 50000$ sur les instances réelles.

Nous remarquons que notre algorithme présente une bonne convergence vers des solutions de même coût. En effet, le caractère aléatoire du choix des opérateurs ne garantit pas la même solution finale. Il permet au contraire de diversifier le parcours des solutions. En dépit de ce caractère aléatoire, l'ALNS donne des solutions qui varient de 1.14% par rapport à A_{25000}et de 0.39% par rapport à A_{50000}.

Impact de la procédure de découpage Nous testons également les modèles de découpage de la phase $\mathcal{P}_1$: (i) modèle avec rotations simples et (ii) modèle avec rotations composées.

Le tableau 7.25 présente le résultat du découpage sur les instances réelles. Nous remarquons que le découpage utilisant les rotations composées présente un meilleur coût systématiquement. Nous remarquons également que moins de véhicules de faible capacité sont utilisés.

	rotations simples				rotations composées			
Instance	coût	SEMI38	8X4	6X4	coût	SEMI38	8X4	6X4
OS22	1653.56	10	12	0	1531.67	12	10	0
OS32	2978.00	21	9	2	2761.47	22	9	1
OS49	4286.45	39	9	1	3890.57	40	8	1
OU35	3217.86	34	1	0	2153.94	35	0	0
OU51_1	4078.51	51	0	0	3616.99	51	0	0
OU51_2	3461.23	51	0	0	3283.40	51	0	0
OU85	4040.01	84	1	0	3446.23	85	0	0

Tableau 7.25 – Ce tableau présente les résultats des deux découpages sur les instances réelles. Pour chaque découpage, nous présentons le coût de la phase de découpage $\mathcal{P}_1$, le nombre de requêtes affectées à la catégorie SEMI38, la catégorie 8X4 et la catégorie 6X4.

Par la suite, nous avons lancé l'ALNS sur ces mêmes instances en considérant deux découpages différents des demandes en utilisant les mêmes paramètres que ceux du tableau 7.22.

Le tableau 7.26 présente ces résultats sur les instances réelles.

Instance	rotations simples A_{25000}	rotations composées A_{25000}
OS22	**2779.92**	2804.17
OS32	**4840.26**	4841.54
OS49	**6384.11**	6387.28
OU35	3276.03	**3264.27**
OU51_1	**5374.81**	**5374.81**
OU51_2	**4985.63**	4996.26
OU85	**5902.89**	5907.01

Tableau 7.26 – Ce tableau présente les résultats de l'ALNS en fonction des deux découpages sur les instances réelles. Nous présentons la meilleure solution obtenue BKS pour chaque découpage.

Pour chaque instance, ces résultats nous montrent que la différence de coût entre les deux versions n'est pas significative. Ces résultats ne nous permettent pas de conclure suffisamment sur l'intérêt d'une méthode de découpage par rapport à une autre.

7.6 Conclusion

Dans ce chapitre, nous avons présenté une méthode permettant de résoudre le problème FT-PDP-RS. Des tests de réalisabilité en temps constant ont été introduits pour permettre des insertions plus rapides dans

notre algorithme ALNS.

Nous avons comparé notre méthode avec des instances de la littérature et celle-ci offre de meilleurs résultats en moyenne et trouve de meilleures solutions quelquefois. Ces résultats montrent, dans un premier temps, l'efficacité de l'algorithme ALNS sur ce problème.

Dans un second temps, nous avons résolu des instances industrielles présentant des contraintes de synchronisation aux ressources. Pour cela, nous avons proposé de nouveaux opérateurs pour parcourir l'espace de solutions. Le problème semble plus difficile à résoudre étant donné la complexité des contraintes introduites. Néanmoins, les instances industrielles sont résolues sous une heure de temps calcul et les solutions retournées sont assez uniformes (peu de déviation).

Les solutions trouvées sont de bonne qualité et meilleures que les solutions retournées par un solveur. Ceci justifie l'utilisation d'une méthode métaheuristique pour la résolution du problème FT-PDP-RS.

8

Résolution du Rich FT-PDP-RS

Sommaire

Ce chapitre traite de l'intégration des contraintes temporelles relatives aux temps de travail des chauffeurs et à la pause-déjeuner. Dans la section 8.1, nous détaillons la méthode retenue pour traiter le problème de minimisation de la durée des tournées. L'intégration des pauses-déjeuner dans la solution sera étudiée dans la section 8.2.

8.1 Minimisation de la durée des tournées

Dans le contexte d'une utilisation industrielle de notre algorithme, le temps de conduite des chauffeurs est une donnée cruciale dans la gestion des ressources humaines. En effet, au-delà des contraintes légales de temps de conduite des chauffeurs, il existe un véritable intérêt en terme d'optimisation à produire des solutions où les chauffeurs conduisent le moins longtemps à service équivalent.

Nous présentons, dans la section 8.1.1, une description du problème de minimisation de la durée des tournées. Dans la section 8.1.2, nous présentons une revue de la littérature sur les problèmes similaires. Dans la section 8.1.3, nous proposons des méthodes pour répondre à cette problématique. Les méthodes seront

évaluées sur les instances et les résultats associés présentés dans la section 8.1.5. Enfin, des perspectives sont présentées dans la section 8.1.6.

8.1.1 Présentation du problème

Dans notre problème, chaque chauffeur effectue une tournée composée d'une succession de points de collectes et de points de livraisons, sur une période d'un horizon temporel. Cette tournée est composée des éléments suivants :

- trajets entres les différents sites ;
- service aux différents sites : chargement et déchargement du camion ;
- temps d'attente aux différents sites.

Pour une tournée k donnée, la durée de la tournée, notée $routeDuration_k$, est donnée par l'expression suivante :

$$routeDuration_k = h_{o_2(k)} - h_{o_1(k)} \tag{8.1}$$

L'heure d'arrivée d'un sommet $i \in V$ est notée a_i et l'heure de début de service h_i. Nous introduisons une variable continue et positive w_i qui représente le temps d'attente à un sommet $i \in V$ entre l'heure d'arrivée et l'heure de début de service. Le chemin entre les nœuds i et j est noté $\psi_{i,j}$.Avec cette variable, l'équation 8.1 est équivalente à l'équation suivante 8.2 :

$$routeDuration_k = \sum_{i \in \psi_{o_1(k),o_2(k)}} s_i + \sum_{(i,j) \in A_k} t_{ij}^k + \sum_{i \in \psi_{o_1(k),o_2(k)}} w_i \tag{8.2}$$

Au regard de cette équation, minimiser la durée d'une tournée est équivalent à minimiser la durée des temps d'attente sur la tournée. En effet, pour une tournée donnée, les temps de trajets (terme t_{ij}^k) et les temps de service (terme s_i) sont constants. Le problème FT-PDP-RS inclue des fenêtres de temps, dès lors, cette minimisation est contrainte par les fenêtres de temps de service des différents points de collecte et livraison.

Pour un problème de tournées de véhicules avec fenêtres de temps, Savelsbergh [93] propose un algorithme efficace permettant d'évaluer en temps constant la durée d'une tournée dans un algorithme de voisinage. Cet algorithme se base sur les *forward time slack*.

Étant donné un ordonnancement au plus tôt, qui est le meilleur choix du point de vue de la réalisabilité des fenêtres de temps, la durée minimale d'une tournée k, notée $routeDuration_k^{min}$ peut être exprimée par l'équation suivante :

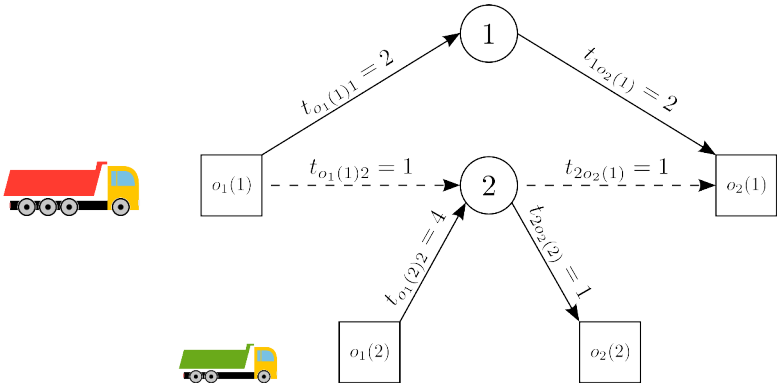
$$routeDuration_k^{min} = h_{o_2(k)} - \left(e_{o_1(k)} + \min(F_{o_1(k)}, \sum_{p=o_1(k)}^{o_2(k)} w_p) \right) \tag{8.3}$$

Cette minimisation revient à décaler l'heure du début de la tournée d'une certaine quantité sans changer l'heure de fin du dernier point de la tournée. Cette quantité est le minimum entre le *forward time slack* $F_{o_1(k)}$ au dépôt d'origine $o_1(k)$d'une tournée k et la somme des temps d'attente sur cette tournée.

Si cette méthode est optimale dans les problèmes de tournées de véhicules avec fenêtres de temps, elle ne l'est pas pour autant en présence de contraintes de synchronisation entre plusieurs tournées. Pour illustrer ce propos, nous présentons un contre-exemple dans la figure 6.

Dans l'exemple 6, nous nous intéressons à la minimisation de la somme des durées des deux tournées 1 et 2. Les fenêtres de temps sur chaque sommet sont mentionnées dans les tableaux respectifs.

Un ordonnancement au plus tôt de ce graphe est donné dans le tableau 8.1. Ce tableau présente l'heure d'arrivée a_i, de début de service h_i, le temps d'attente w_i ainsi que le *forward time slack* F_i à chaque point i. La somme des durées des tournées est de 15 ($routeDuration_1 = 10$ et $routeDuration_2 = 5$).



Exemple 6 – Cette figure montre un contre-exemple pour la minimisation des tournées. Deux véhicules (1 et 2) effectuent leur tournée. Le véhicule 1 effectue le chemin $(o_1(1), 1, o_2(1))$ et le véhicule 2 le chemin $(o_1(2), 2, o_2(2))$. Le temps de trajet entre les sommets est mentionné sur l'arc. Les tournées des véhicules sont symbolisées par les traits pleins. Il existe une synchronisation symbolisée par les traits pointillés entre les véhicules au niveau du sommet 2.

i	e_i	l_i	a_i	h_i	w_i	F_i
$o_1(1)$	0	10	0	0	0	5
1	0	10	2	2	0	6
$o_2(1)$	10	10	4	10	6	0
$o_1(2)$	0	10	0	0	0	5
2	0	10	4	4	0	5
$o_2(2)$	0	10	5	5	0	5

Tableau 8.1 – Ordonnancement au plus tôt de l'exemple 6.

Dès lors, en suivant le principe énoncé par Savelsbergh, nous pouvons chercher à décaler l'heure de début des sommets $o_1(1)$ et $o_1(2)$ pour proposer un ordonnancement de durée minimale. La solution résultante est présentée dans le tableau 8.2. Cette solution présente un meilleur objectif de valeur 12 ($routeDuration_1 = 7$ et $routeDuration_2 = 5$).

i	e_i	l_i	a_i	h_i	w_i
$o_1(1)$	0	10	3	3	0
1	0	10	5	5	0
$o_2(1)$	10	10	7	10	3
$o_1(2)$	0	10	0	0	0
2	0	10	4	4	0
$o_2(2)$	0	10	5	5	0

Tableau 8.2 – Ordonnancement au plus tôt avec décalage de l'heure de départ aux sommets $o_1(1)$ et $o_1(2)$ de l'exemple 6.

Si cette méthode est optimale pour un problème de tournées de véhicules sans synchronisation, elle ne l'est pas dans notre cas où il existe une synchronisation entre les deux tournées au point 2. En effet, en autorisant de terminer plus tard aux derniers sommets $o_2(1)$ et $o_2(2)$, il existe une meilleure solution qui minimise la somme des durées des tournées. Cette solution est présentée dans le tableau 8.3. Cet ordonnancement propose une solution de coût 9 ($routeDuration_1 = 4$ et $routeDuration_2 = 5$) avec aucun temps d'attente.s

i	e_i	l_i	a_i	h_i	w_i
$o_1(1)$	0	10	6	6	0
1	0	10	8	8	0
$o_2(1)$	10	10	10	10	0
$o_1(2)$	0	10	5	5	0
2	0	10	9	9	0
$o_2(2)$	0	10	10	10	0

Tableau 8.3 – Ordonnancement optimal de l'exemple 6.

Au regard de cet exemple et avec l'objectif de répondre à notre problématique, nous définissons le problème de minimisation de la durée des tournées lorsque celles-ci sont synchronisées. Étant donné un ensemble de tournées synchronisées comportant chacune une séquence fixe d'opérations (collectes et livraisons), il s'agit de trouver l'heure de début de service de chaque opération ainsi que l'heure de début et fin de chaque tournée. Chaque opération est ordonnancée sur une ressource qui peut traiter une et une seule opération à la fois.

Ce problème peut être vu comme un problème d'ordonnancement de projet où le critère à minimiser est la somme des temps d'attentes $\sum_{i \in V} w_i$.

8.1.2 Littérature

Nous proposons une revue de la littérature pour caractériser le problème que nous souhaitons résoudre.

Dans le champ des problèmes de tournées de véhicules, les travaux de Savelsbergh [93] ont été précurseurs pour la recherche de voisinage local dans les problèmes comportant des fenêtres de temps. Les auteurs se sont penchés sur la production de solutions où les temps d'attente pouvaient être réduits dans l'objectif de diminuer la durée d'une tournée. Dans les méthodes heuristiques à recherche de voisinage, l'ordonnancement dans le but de trouver des tournées de durée minimale est un problème difficile à cause de

l'interdépendance entre les tournées. Les opérateurs d'insertions nécessitent souvent le calcul d'un nouvel ordonnancement des tournées pour évaluer leurs coûts. Comme expliqué dans l'état de l'art de Drexl [35], trois méthodes sont proposées pour réaliser à ce calcul : (i) permettre l'irréalisabilité, (ii) avoir une approximation du coût d'une insertion et (iii) explorer un espace de recherche auxiliaire où des contraintes du problème initial sont relâchées [31].

Les méthodes utilisées dans les problèmes d'ordonnancement sont de bons candidats pour notre problème. Suivant la classification de Brucker [19], notre problème peut être classé de plusieurs façons.

Premièrement, ce problème peut être vu comme un problème de Job Shop. Ils existent n tournées (*jobs*) qui doivent être réalisées sur m ressources (*machines*). Chaque tournée i est composée de n_i opérations qui doivent être faites dans un ordre déterminé (contraintes de précédence entre les opérations). À chaque opération sont associés une ressource et un temps d'exécution. Dans notre cas, le problème comporte m ressources différentes. La préemption des opérations n'est pas autorisée et chaque opération a une date de disponibilité (*release date*) et une date d'échéance (*due date*). Un problème similaire a été étudié par Chu et Portmann [21]. Pour minimiser le temps d'attente total, une heuristique à base de règles de priorité est développée et améliore de 3% en moyenne les résultats par rapport à une heuristique basée sur la règle du plus court temps de service (*Shortest Processing Time*). Les auteurs suggèrent d'utiliser les deux heuristiques pour résoudre un problème réel. C'est le seul travail de recherche proche de notre problème à notre connaissance.

Notre problème peut également être vu comme un problème de Flow Shop Hybride (HFS). Le graphe temporel G_t peut être représenté par niveaux. Chaque niveau correspond alors à une étape du HFS. À chaque niveau, un ensemble d'opérations (collecte ou livraison) doit être effectué. Certaines opérations peuvent ne pas être effectuées à un stage donné. Un état de l'art de Ruiz et Vazquez-Rodriguez [89] classe le problème HFS dans les problèmes **NP**-difficile. Les auteurs montrent, sur une étude de plusieurs centaines de papiers, que les méthodes de résolution à base d'heuristiques sont largement majoritaires (90% des papiers étudiés) et 60% des papiers de cet état de l'art sont dédiés à la minimisation du *makespan*.

Nous avons cherché dans la littérature un problème similaire au nôtre mais nous n'en avons pas trouvé. La littérature des problèmes d'ordonnancement est très vaste et beaucoup de problèmes étudiés comportent des variantes dans leur définition qui ne permettent pas de se comparer à notre problème. Les problèmes d'ordonnancement les plus étudiés sont ceux dont le critère à minimiser est dit régulier [73], c'est à dire qu'il est une fonction croissante de l'heure de fin de réalisation des opérations, en d'autres mots, il est toujours meilleur de commencer les opérations le plus tôt possible. Les méthodes classiques de la littérature ne permettent pas de minimiser correctement la durée des tournées sur un problème avec synchronisation. Nous proposons donc une approche pour traiter ce problème.

8.1.3 Modélisation du problème de minimisation de la durée des tournées

Nous présentons la modélisation du problème de minimisation de la durée des tournées. Nous verrons que ce problème comporte plusieurs objectifs i notés z_i^{route}.

Dans le graphe temporel G_t défini dans la section 7.3.3, chaque précédence entre deux sommets consécutifs d'une tournée ou bien deux sommets consécutifs sur une ressource est modélisée par un arc dans G_t. L'ensemble des arcs appartenant aux tournées est noté $A_K \subset A$ et l'ensemble des arcs appartenant aux ressources est noté $A_\Pi \subset A$. Nous avons également $A = A_K \cup A_\Pi$ avec $A_K \cap A_\Pi = \emptyset$. Le poids de l'arc entre les sommets i et j est noté t_{ij}^k et est égal au temps de trajet entre ces deux sommets par le véhicule $k \in K$. À un sommet i, h_i représente l'heure de début de service et a_i l'heure d'arrivée. Nous définissons par C_{max} l'heure au plus tard de fin de toutes les tournées.

Le premier objectif z_1^{route} de notre problème n'est pas un critère régulier tel que le C_{max}. Dans un premier temps, nous cherchons à produire des tournées de durée minimale, c'est à dire $z_1^{route} = \min \sum_{k \in K} routeDuration_k$. L'équation 8.2 nous montre que ceci est équivalent à minimiser la somme des temps d'attente, c'est à dire $z_1^{route} = \min \sum_{i \in V} w_i$.

De plus, il est préférable que les chauffeurs effectuent leurs tournées en commençant tôt et en finissant

tôt. Pour cela, nous définissons un deuxième objectif z_2^{route} où nous minimisons la fin de tournée maximum, appelée *makespan* : $z_2^{route} = \min C_{max}$. Cet objectif est fortement lié à une autre volonté de l'entreprise qui est de fournir des plannings équilibrés de chauffeurs.

Enfin, il peut être intéressant de produire des plannings équilibrés. Pour cela, nous définissons un troisième objectif z_3^{route} qui minimise la somme des heures de fin de tournées : $z_3^{route} = \min \sum_{k \in K} h_{o_2(k)}$.

Cette proposition d'objectifs repose à la fois sur les besoins de l'entreprise quant à la production des plannings des chauffeurs et l'analyse de l'exemple 6 qui montre que l'heure de début peut avoir une influence sur la durée minimale des tournées (cf. tableau 8.2 et tableau 8.3).

Ces trois termes sont pris en compte de façon lexicographique dans la fonction objectif. Nous définissons l'ordre lexicographique suivant pour toute solution s : $z_1^{route}(s) \leq_{\text{lex}} z_2^{route}(s) \leq_{\text{lex}} z_3^{route}(s)$. Nous appelons ce problème minimisation de la durée des tournées, abrégé en $\mathcal{P}_{MINDUR}$. Sa modélisation sous forme de programme linéaire est la suivante :

$$\min \ z_1^{route} = \sum_{k \in K} (h_{o_2(k)} - h_{o_1(k)}) \tag{8.4}$$

$$\min \ z_2^{route} = C_{max} \tag{8.5}$$

$$\min \ z_3^{route} = \sum_{k \in K} h_{o_2(k)} \tag{8.6}$$

$$s.t.$$

$$a_i \leq h_i \qquad \forall i \in V \tag{8.7}$$

$$h_i \leq C_{max} \qquad \forall i \in V \tag{8.8}$$

$$h_i + t_{ij} + s_i = a_j \qquad \forall (i,j) \in A_K \tag{8.9}$$

$$h_i + s_i \leq h_j \qquad \forall (i,j) \in A_\Pi \tag{8.10}$$

$$a_i, C_{max} \in \mathbb{R}^+ \qquad \forall i \in V \tag{8.11}$$

$$h_i \in [e_i, l_i] \qquad \forall i \in V \tag{8.12}$$

Les fonctions objectif à minimiser sont définies par les équations (8.4),(8.5) et (8.6). Les contraintes (8.7) imposent l'heure d'arrivée avant l'heure de service à chaque sommet. Les contraintes (8.8) lient les variables C_{max} et h. Les précédences sur les tournées sont modélisées par les contraintes (8.9) et sur les ressources par les contraintes (8.10).

8.1.4 Intégration dans l'ALNS

Modification de la fonction objectif

Pour inclure la perspective de durée des tournées, nous modifions la fonction objectif de l'ALNS en incluant la durée totale des tournées. Cela revient à inclure le coût des temps d'attente dans la fonction objectif du problème f' et de surcroît la fonction objectif pénalisée f'_p.

$$\begin{aligned} f' = \min & \sum_{k \in K} \mathcal{CD}^k * y^k \\ & + \sum_{k \in K} \sum_{(i,j) \in A} \mathcal{CK}^k * x_{ij}^k * d_{ij}^k \\ & + \sum_{k \in K} routeDuration_k \end{aligned} \tag{8.13}$$

$$
\begin{aligned}
f'_p = \min & \sum_{k \in K} \mathcal{CD}^k * y^k \\
& + \sum_{k \in K} \sum_{(i,j) \in A} \mathcal{CK}^k * x^k_{ij} * d^k_{ij} \\
& + \sum_{k \in K} routeDuration_k \\
& + \kappa \sum_{k \in K} \sum_{r \in R} (1 - u^k_r)
\end{aligned} \tag{8.14}
$$

Cette modification est faite en utilisant le résultat de l'équation 8.2.

Résolution du modèle par un solveur de programmation linéaire

Le problème $\mathcal{P}_{MINDUR}$ est un problème de type PL avec plusieurs objectifs à minimiser. Une possibilité de résolution est d'agréger les trois objectifs dans une somme pondérée. Nous pouvons le faire aisément puisque tous les objectifs sont exprimés en fonction de variables temporelles de même unité.

La considération multiobjectif [42] de ce problème n'est pas prise en compte dans le cadre de cette thèse. Nous souhaitons considérer les objectifs de façon lexicographique. Nous proposons trois paramètres $\alpha^{route}_1, \alpha^{route}_2, \alpha^{route}_3$ pour pondérer les trois objectifs avec $\alpha^{route}_1 \gg \alpha^{route}_2 \gg \alpha^{route}_3$ et $\sum_{i=1}^{3} \alpha^{route}_i \geq 1$.

La définition de cette pondération permet de considérer les objectifs dans l'ordre lexicographique relativement à leurs poids et modélise fidèlement les objectifs de l'entreprise.

Cette modélisation nous permet d'utiliser un solveur de PL pour résoudre $\mathcal{P}_{MINDUR}$. La fonction objectif du problème $\mathcal{P}_{MINDUR}$ résolue par le solveur est définie par $f^{route} = \sum_{i=1}^{n} \alpha^{route}_i * z^{route}_i$.

L'appel à la méthode SOLVEPMINDUR permet de résoudre le problème $\mathcal{P}_{MINDUR}$.

Résoudre $\mathcal{P}_{MINDUR}$ à chaque évaluation d'insertion n'est pas envisageable pour des raisons de temps de calcul liés au chargement de l'instance par le solveur principalement. Aussi, nous proposons deux approches :

- utiliser la fonction objectif du FT-PDP-RS pour guider la recherche et résoudre le PL de manière sélective au moment de l'acceptation de la solution ;
- estimer en temps constant l'augmentation de la durée des tournées d'une solution dans les opérateurs d'insertion.

Modification de la procédure de sélection d'une meilleure solution

Dans l'algorithme de l'ALNS, une solution nouvelle s_{new} est acceptée comme nouvelle meilleure solution s_{best} si elle est de meilleur coût (algorithme 1 ligne 13). Pour effectuer cette comparaison, dans la méthode standard, nous utilisons la fonction de coût pénalisée du problème définie dans la section 7.2.1.

La procédure standard de sélection est rappelée dans l'algorithme 11.

Algorithme 11 : ACCEPTATION D'UNE SOLUTION (PROCÉDURE STANDARD)

Entrées : une solution nouvelle s_{new}, la meilleure solution s_{best}

```
1 début
2     si f_p(s_new) ≤ f_p(s_best) alors
3         s_best ← s_new
4     fin
5 fin
```

Nous proposons de modifier cette méthode d'acceptation. Si une solution s_{new} est strictement meilleure que la solution s_{best}, selon la fonction objectif pénalisée f'_p, alors elle est acceptée comme nouvelle meilleure solution. Si elle est proche à une distance α, alors le problème $\mathcal{P}_{MINDUR}$ est résolu sur les deux solutions et si la solution s_{new} est meilleure par rapport aux z_i^{route}, elle est acceptée comme nouvelle meilleure solution s_{best}. Cette méthode d'acceptation est présentée dans l'algorithme 12.

Algorithme 12 : ACCEPTATION DE LA SOLUTION (PROCÉDURE MODIFIÉE)

Entrées : une solution nouvelle s_{new}, la meilleure solution s_{best}

début
  si $f'_p(s_{new}) < f'_p(s_{best})$ **alors**
    $s_{best} \leftarrow s_{new}$
  sinon si $f'_p(s_{new}) \leq (1+\alpha) f'_p(s_{best})$ **alors**
    solvePMINDUR(s_{new})
    si $f^{route}(s_{new}) < f^{route}(s_{best})$ **alors**
      $s_{best} \leftarrow s_{new}$
    fin
  fin
fin

Nous étudions par la suite l'ajout de cette nouvelle procédure d'acceptation de la meilleure solution dans notre algorithme ALNS.

Nouvel opérateur d'insertion pour la minimisation des tournées

Dans le but de favoriser la minimisation de la durée des tournées, nous proposons un nouvel opérateur d'insertion qui estime l'augmentation de la durée des tournées dans l'évaluation du coût d'une insertion. Nous nous intéresserons principalement, par la suite, à la prise en compte du critère z_1^{route} où nous cherchons à minimiser les temps d'attente.

Dans les opérateurs standards d'insertion de l'ALNS, l'évaluation de l'insertion d'une requête utilise le coût lié à l'augmentation du temps de trajet et l'augmentation de la distance. Ce coût ne prend pas en compte le coût lié à l'augmentation des temps d'attentes potentiellement créés aux successeurs de cette requête.

La figure 8.1 représente l'insertion d'une requête r, composée du point de collecte p_r et du point de livraison d_r, entre les sommets i et σ_i sur la tournée k. Cette insertion va potentiellement décaler les successeurs de la requête. Nous appelons δ_i le décalage temporel du sommet i après l'insertion de la requête. Soit h'_i l'heure de début de service de i, nous avons $\delta_i = h'_i - h_i$. Insérer la requête r avec le point de collecte p_r et le point de livraison d_r après le point i, p_r après u, d_r après v, peut augmenter l'heure de service des sommets $\sigma_i, \sigma_u, \sigma_v$.

Cette insertion crée un décalage temporel potentiel aux sommets σ_i, σ_u et σ_v. De plus, l'inter-dépendance des tournées peut occasionner le décalage temporel du sommet du dépôt d'arrivée de la tournée p de la figure 8.1 et des dépôts d'arrivée des autres tournées. En résultat, la solution peut avoir des temps d'attente supplémentaires aux différents sommets et les heures de fin des tournées peuvent être décalées.

Pour un ordonnancement au plus tôt d'une tournée, les travaux de Savelsbergh [93] ont montré qu'il est possible de trouver une tournée de durée minimale, sans décaler l'heure d'arrivée, en décalant l'heure de début de tournée d'une certaine quantité en utilisant les *forward time slack*. Ceux-ci sont calculés, dans une tournée, en connaissant l'heure de service de chaque sommet.

Dans notre problème, une insertion peut décaler l'heure de service des sommets situés en fin de tournée. Cette insertion implique que les *forward time slack* de notre solution ne sont plus valides après l'insertion d'une requête. Les travaux de Savelsbergh ne sont plus applicables.

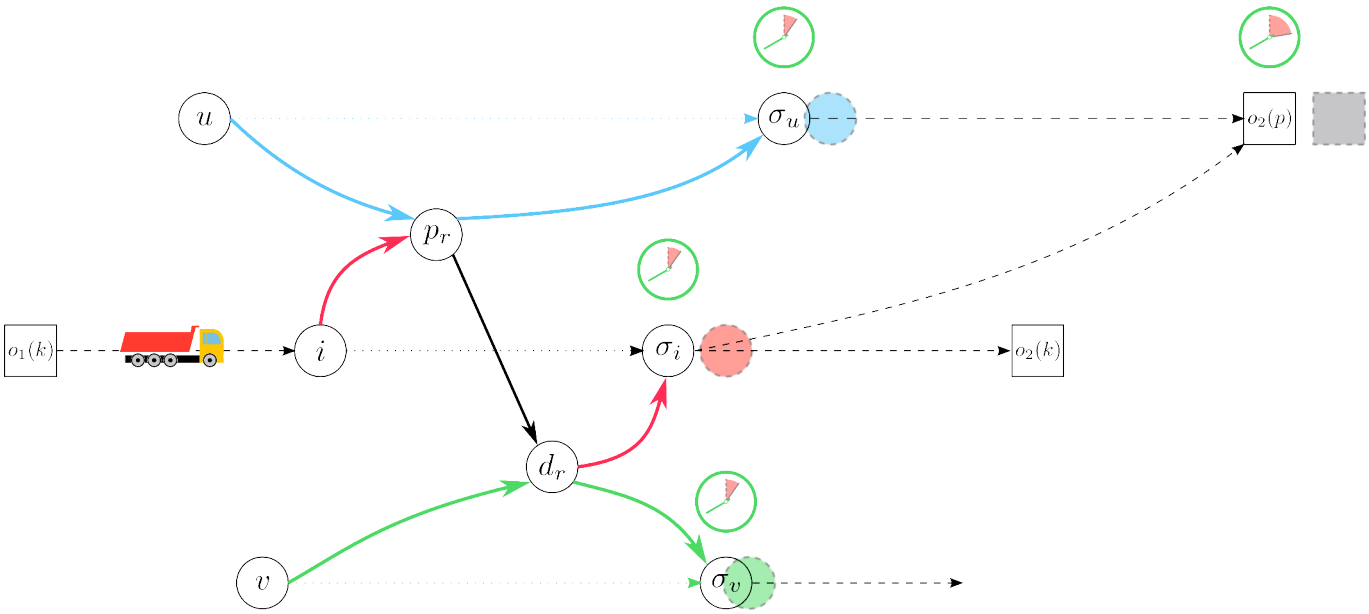


FIGURE 8.1 – Cette figure montre le décalage pouvant se produire aux dépôts d'arrivée de certaines tournées lors de l'insertion d'une requête r après le point i.

Les travaux de Masson et al. [71] ont introduit la notion de *slack time* qui définit la somme des temps d'attente du plus court chemin entre deux sommets. La quantité $ST_{i,o_2(k)}$ représente le *slack time* ou la somme des temps d'attente entre le sommet i et la fin de tournée du véhicule k. Le terme $max(0, \delta_i - ST_{i,o_2(k)})$ représente l'augmentation de temps au sommet $o_2(k)$ par le décalage du sommet i. Cette augmentation est liée au décalage de δ_i qui peut être absorbée par une partie des temps d'attente.

Dans notre cas, l'heure de service du dépôt d'arrivée d'une tournée peut être décalée suite aux décalages, non pas d'un sommet, mais de trois sommets : (i) σ_i, (ii) σ_u et (iii) σ_v. Nous nous posons alors deux questions :

- Pouvons-nous quantifier l'augmentation des temps d'attente à chaque sommet ?
- Pouvons-nous quantifier l'augmentation de la durée des tournées ?

Il est intéressant alors de pouvoir quantifier ces augmentations. À notre connaissance, il n'existe pas de travaux ou d'algorithmes proposant une approche à ces questions. Pour cela, nous proposons une heuristique d'évaluation de cette augmentation en se basant sur les travaux de Savelsbergh [93] et de Masson et al. [71].

L'insertion de la requête r, présentée dans la figure 8.1, décale les successeurs directs des sommets p_r et d_r à savoir : σ_i, σ_u et σ_v. Les décalages respectifs sont : δ_{σ_i}, δ_{σ_u} et δ_{σ_v}.

Nous considérons un ordonnancement au plus tôt dans le graphe G_t. Soit $\Delta_i^{\delta_j}$ le décalage temporel à un point i provoqué par le décalage du point j. Le décalage du nœud de fin d'une tournée $p \in K$ peut être causé par 3 décalages différents :

- par le décalage de σ_i : $\Delta_{o_2(p)}^{\delta_{\sigma_i}} = max(0, \delta_{\sigma_i} - ST_{\sigma_i,o_2(p)})$;
- par le décalage de σ_u : $\Delta_{o_2(p)}^{\delta_{\sigma_u}} = max(0, \delta_{\sigma_u} - ST_{\sigma_u,o_2(p)})$;
- par le décalage de σ_v : $\Delta_{o_2(p)}^{\delta_{\sigma_v}} = max(0, \delta_{\sigma_v} - ST_{\sigma_v,o_2(p)})$.

Les tournées étant interdépendantes, certains de ces décalages peuvent s'annuler entre eux. En effet, la réalisation d'un des décalages provoque le changement des heures de fin de tournées à cause des *slack time* entre chaque paire de nœuds. Pour cela, nous proposons de compter que la plus défavorable des augmentations pour estimer, $\Delta_{o_2(p)}$, le décalage temporel du nœud de fin d'une tournée $p \in K$.

$$\Delta_{o_2(p)} = max(\Delta^{\delta_{\sigma_i}}_{o_2(p)}, \Delta^{\delta_{\sigma_u}}_{o_2(p)}, \Delta^{\delta_{\sigma_v}}_{o_2(p)}) \tag{8.15}$$

Enfin, pour estimer le décalage total Δ lié à l'insertion d'une requête, nous prenons la somme des décalages pour toutes les tournées.

$$\Delta = \sum_{k \in K} \Delta_{o_2(k)} \tag{8.16}$$

Cette approximation du décalage nous sert à estimer le coût d'une insertion d'une requête en prenant en compte l'augmentation de la durée des tournées, donc l'augmentation des temps d'attente. Le coût estimé, $\Delta^{i,u,v}$, d'une insertion de la requête r après le sommet i dans la tournée $k \in K$, le sommet u dans la ressource de la collecte et le sommet v dans la ressource de la livraison, est donné par l'équation 8.17.

$$\begin{aligned}\Delta^{i,u,v} = & \mathcal{CK}^k * (d_{ip_r} + d_{d_r\sigma_i} + d_{p_rd_r} - d_{i\sigma_i}) \\ & + \mathcal{CH}^k * (t_{ip_r} + t_{d_r\sigma_i} + t_{p_rd_r} - t_{i\sigma_i}) \\ & + \mathcal{CH}^k * \Delta\end{aligned} \tag{8.17}$$

À la différence du *framework* des opérateurs d'insertion, algorithme 8, proposé dans la section 7.2.6, nous devons considérer un nouveau concept d'insertion. En effet, dans le *framework* précédent, nous considérions, en deux temps, (i) le choix d'une position dans la tournée et (ii) le choix d'une position dans les ressources ; l'évaluation du coût d'une insertion s'effectuant uniquement avec le choix (i).

La nouvelle évaluation proposée par l'équation 8.17 impose la connaissance exacte du positionnement de la requête dans la tournée et aux ressources. Pour cela, nous proposons une nouvelle heuristique de réparation **HMINDUR** dont le fonctionnement est le suivant :

1. sélection d'une requête (méthode $\mathrm{RequestSelection}$) ;
2. insertion de la requête dans une tournée et une ressource (méthode $\mathrm{RequestInsertionRouteResource}$).

La sélection d'une requête r s'effectue avec les mêmes opérateurs de sélection que le *framework* standard. Toutes les possibilités, dans les tournées après le sommet i et dans les ressources après les sommets u et v, d'insertion pour cette requête r sont calculées et la meilleure possibilité $(i_{best}, u_{best}, v_{best})$, celle qui minimise le critère $\Delta^{i,u,v}$, est effectuée.

Nous considérons une solution partielle $s_{partial}$ dans laquelle des requêtes ne sont pas servies ($\mathcal{R} \neq \emptyset$). L'heuristique est proposée dans l'algorithme 13.

Nous étudions par la suite l'ajout de ce nouvel opérateur de réparation dans notre algorithme ALNS.

8.1.5 Expérimentations numériques

Étude de la contribution de la nouvelle procédure de sélection de la meilleure solution

Nous étudions tout d'abord la procédure d'acceptation d'une meilleure solution décrite dans l'algorithme 12. Nous allons étudier le comportement du paramètre α. Pour cela, nous établissons le même protocole de tests que dans la section 7.5. Nous allons tester quatre valeurs du paramètres $\alpha = \{0.01, 0.05, 0.10, 0.15\}$ sur trois instances réelles (OS22,OS32,OU51_1).

Les tests ont été exécutés avec les paramètres listés dans le tableau 8.4.

Le tableau 8.5 présente le pourcentage moyen de déviation à la meilleure solution connue lors de ces tests.

L'analyse des résultats montre que l'augmentation du paramètre α diminue la moyenne de la déviation à la meilleure solution. Nous choisissons d'opter pour le paramètre $\alpha = 0.01$. Cette valeur du paramètre

Algorithme 13 : Heuristique **HMINDUR** la minimisation de la durée des tournées.

Entrées : une solution partielle $s_{partial}$, un ensemble de requêtes non insérées $\mathcal{R}$
Sortie : une solution complète

```
1  début
2      tant que plus d'insertion réalisable faire
           /* Sélection d'une requête à insérer                          */
3          r ← RequestSelection(R)
4          Δ_best ← +∞
5          (i_best, u_best, v_best) ← null
           /* Calculer le critère pour toutes les positions (i, u, v)    */
6          pour toutes les positions (i, u, v) faire
7              Δ ← RequestInsertionRouteResource(r)
8              si Δ^{i,u,v} < Δ_best alors
9                  (i_best, u_best, v_best) = (i, u, v)
10             fin
11         fin
           /* Insertion r dans la solution                               */
12         s ← RequestInsertionRouteResource(r)
13         ResourceScheduling(r)
14         R ← R − {r}
15     fin
16 fin
```

Paramètre	Valeur
Solution initiale	**2R-CI-EF**
κ	100000
t_{limit}	$600s$
bruitage	oui
σ_1	33
σ_2	20
σ_3	13
ξ_{min}	10
ξ_{max}	20
c	0.99975

Tableau 8.4 – Présentation des paramètres retenus pour l'exécution des tests sur le paramètre α.

Instance	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.10$	$\alpha = 0.15$
OS22	**1.80**	2.04	2.19	2.17
OS30	0.40	0.38	0.40	**0.36**
OU51_1	1.76	**1.74**	2.42	2.76
moy.	**1.32**	1.38	1.67	1.77

Tableau 8.5 – Résultats pour quatre configurations du paramètre α : le tableau présente le pourcentage moyen de déviation à la meilleure solution connue avec $t_{limit} = 600s$.

permet d'accepter des solutions comme meilleure solution de coût supérieur à 1% de la meilleure solution mais qui présente un meilleur coût lié à la fonction f^{route}.

Nous testons maintenant cette nouvelle procédure d'acceptation de la meilleure solution sur toutes les instances réelles en considérant les paramètres présentés dans le tableau 8.6.

Paramètre	Valeur
Solution initiale	**2R-CI-EF**
κ	100000
t_{limit}	$3600s$
Φ	25000
τ	5000
bruitage	oui
σ_1	33
σ_2	20
σ_3	13
ξ_{min}	10
ξ_{max}	20
α	0.01
c	0.99975

Tableau 8.6 – Présentation des paramètres retenus pour l'exécution les tests de la nouvelle procédure d'acceptation de la meilleure solution.

Le tableau 8.7 présente les résultats sur les instances réelles de l'utilisation de la nouvelle procédure de sélection de la meilleure solution et l'ajout de l'opérateur **HMINDUR**. Nous comparons la fonction objectif f puisque c'est la fonction objectif ne comportant pas les temps d'attente. Elle est une base de comparaison pour les deux problèmes. Nous montrons aussi les temps d'attente présents dans la meilleure solution.

	Fonction objectif f			Temps d'attente w		
Instance	A_{25000}	A'_{25000}	$GAP_{A_{25000}}$ (%)	$w_{A_{25000}}$	$w_{A'_{25000}}$	$GAP_{w_{A_{25000}}}$ (%)
OS22	2779.92	2804.17	0.87	154.98	14.31	-90.77
OS32	4840.26	4944.11	2.15	213.50	17.22	-91.94
OS49	6384.11	6407.26	0.36	346.65	311.05	-10.27
OU35	3276.03	3276.03	0	205.14	114.10	-44.38
OU51_1	5374.81	5374.81	0	53.12	158.57	198.49
OU51_2	4985.63	5004.77	0.38	259.57	122.06	-52.98
OU85	5902.89	5915.58	0.22	171.46	236.69	38.05
moy.			0.56			-7.68

Tableau 8.7 – Ce tableau présente les résultats de l'ALNS en fonction des deux procédures de sélection de la meilleure solution et l'heuristique **HMINDUR**. Nous présentons la meilleure solution obtenue (fonction f) et les temps d'attente (en min.) dans la meilleure solution.

Ces résultats montrent que les solutions retournées par la prise en compte des temps d'attente ne sont pas meilleures lorsque nous comparons la fonction objectif f. En revanche, les solutions retournées sont meilleures lorsque nous nous intéressons aux temps d'attente. Les solutions retournées sont en moyenne 8% meilleures. Ces tests nous permettent de conclure de l'apport de l'heuristique **HMINDUR** et de la procédure de sélection de la meilleure solution.

8.1.6 Perspectives

Nous avons étudié la minimisation de la durée des tournées pour le problème FT-PDP-RS. Pour cela, nous avons introduit une nouvelle procédure d'acceptation de la meilleure solution pour notre métaheuristique ALNS. Nous avons également développé une nouvelle heuristique d'insertion **HMINDUR** permet d'évaluer l'augmentation des temps d'attente dans le coût d'une insertion.

Le développement de ces méthodes spécifiques à notre problème portant sur la minimisation de la durée des tournées ne nous a pas permis d'améliorer le coût des tournées par rapport à la fonction objectif f. En revanche, les résultats montrent l'apport de ces méthodes lorsque nous souhaitons minimiser le coût réel des tournées incluant le coût des temps d'attente.

Nous avons vu que le problème de minimisation de la durée des tournées est un problème pouvant être classé dans les problèmes d'ordonnancement. La résolution relativement rapide de cet ordonnancement par un solveur de PLNE est un gain non négligeable dans le test de la procédure de sélection de la meilleure solution.

En libérant la contrainte de précédence sur les ressources, un problème différent pourrait être résolu par des méthodes venant de l'ordonnancement. C'est une des perspectives de travail. Nous pouvons facilement revenir à un diagramme PERT dans lequel nous cherchons à minimiser un critère. Le critère de de minimisation des temps d'attente, peu commun dans la littérature, est une possibilité pour des travaux futurs.

8.2 Intégration des pauses-déjeuner

Conjointement à la minimisation de la durée des tournées, la production d'un planning de tournées réaliste et réalisable impose la satisfaction des contraintes du problème. Dans cette section, nous nous intéressons à la satisfaction des contraintes légales liées aux temps de conduite des chauffeurs.

La section 8.2.1 présente une revue de la littérature sur les problèmes de temps de conduite des chauffeurs ainsi qu'une présentation de nos objectifs. Le problème d'intégration des pauses-déjeuner est formulé dans la section 8.2.2. Les algorithmes utilisés pour résoudre ce problème sont décrits dans la section 8.2.3. Les résultats numériques sont présentés dans la section 8.2.4. La section 8.2.5 conclue cette partie.

8.2.1 Etat de l'art et règles pour l'intégration des pauses-déjeuner

Un des premiers papiers concernant la prise en compte des heures de travail des chauffeurs est celui de Goel et Gruhn [49, 48]. Ils proposent une modélisation du problème de VRPTW en intégrant la prise en compte des horaires des chauffeurs. Le problème décrit est le Vehicle Routing Problem with Drivers' Workign Hours (VRPDWH). Ils utilisent des labels pour connaître l'état du chauffeur (conduite, repos, pause, travail) à chaque point d'une tournée. Un algorithme de type LNS résout des instances modifiées de Solomon [100]. À chaque insertion, les labels du client inséré ainsi que ceux des suivants dans la tournée sont recalculés.

Pour la législation européenne des horaires de conduite [79], Prescott-Gagnon et al [83] proposent une méthode LNS pour la construction de tournées pour le Vehicle Routing Problem with Time Windows with Drivers Rules (VRPTWDR). La reconstruction des tournées est réalisée par une heuristique de type *Branch-and-Price* et un algorithme de labels vérifie la réalisabilité des tournées créées. Les auteurs ne proposent pas d'étude de complexité pour l'algorithme de réalisabilité car la législation européenne diffère de la législation des États-Unis (voir l'article de Archetti et Savelsbergh [5]).

Goel et Kok [50] proposent un algorithme en $\mathcal{O}(\lambda)$ avec λ le nombre de clients à visiter qui permet de déterminer les temps de conduite, repos et pauses d'un chauffeur, pour cette même législation européenne.

Peu de travaux concerne la prise en compte des contraintes temporelles des chauffeurs dans la construction de tournées. Le problème de construction de tournées est **NP**-difficile ; rajouter la réalisabilité des

contraintes liées aux temps de conduite des chauffeurs se traduit par un problème au moins aussi difficile (voir [49]).

Dans notre cas, nous nous intéressons à la législation européenne qui prévaut sur la législation française et impose un certain nombre de réglementations liées aux temps de conduite des chauffeurs. Le document [79] regroupe toutes ces réglementations. Parmi toutes les règles pouvant être prises en compte, nous nous intéresserons aux suivantes :

- La durée maximale de conduite journalière est de 9h (T^{daily}) ;
- La durée maximale de conduite continue ne doit pas excéder 4h30 ($T^{interval}$). Le chauffeur devra alors respecter une pause de 45 minutes.
- Le temps de disponibilité ou de déjeuner du midi vaut pour interruption de la conduite continue et est considéré comme une pause.

La réglementation [79] impose également la présence d'un chronotachygraphe numérique sur les véhicules. Ce système permet à chaque chauffeur, au moyen d'une puce personnelle, d'enregistrer toutes les informations liées à la conduite du véhicule (vitesse, temps de conduite, activités, etc.). Une analyse de la conduite est alors fournie par l'appareil à l'issue d'une période d'utilisation. On peut y retrouver les différentes phases de conduite dont les principales sont synthétisées dans le tableau 8.8.

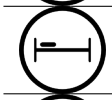
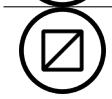
symbole	description
	période de **conduite** du chauffeur
	période durant laquelle le chauffeur réalise une **activité** au service de l'entreprise
	période de **repos** lorsque le chauffeur est libre de ses mouvements et le camion est fermé à clé
	période de **disponibilité** lorsque le chauffeur ne réalise ni une activité ni une conduite

Tableau 8.8 – Symboles présents sur un chronotachygraphe.

Au regard de la complexité des algorithmes de la littérature (souvent au minimum quadratique) et dans le but de maintenir une complexité constante sur les tests de réalisabilité de notre algorithme ALNS, nous décidons d'intégrer les deux premières règles dans nos tests de réalisabilité.

Dans la suite de ce document, nous nous intéressons uniquement à la prise en compte des pauses-déjeuner dans les tournées.

8.2.2 Formulation du problème de l'intégration des pauses-déjeuner

Dans cette section, nous proposons une modélisation du problème d'intégration des pauses-déjeuner. Comme pour la minimisation des tournées, nous nous basons sur une solution existante de notre problème dans laquelle nous cherchons à insérer les pauses-déjeuner.

Nous nommons ce problème $\mathcal{P}_{LUNCH}$. Cette pause-déjeuner permet au chauffeur de séparer sa tournée en deux parties. Cette séparation de la tournée nous permet de plus facilement vérifier la contrainte de conduite continue qui ne doit pas dépasser 4h30.

En effet, nous pouvons stocker dans la mémoire le temps de conduite avant la pause-déjeuner et le temps de conduite après la pause-déjeuner. Lors d'une insertion, nous pouvons tester en temps constant ($\mathcal{O}(1)$) en rajoutant le temps de conduite de la requête insérée, que le nouveau temps de conduite n'excède pas $T^{interval}$.

En se basant sur le graphe temporel G_t, nous considérons les notations suivantes. L'ensemble des sommets représentant des pauses-déjeuner est nommé L. Pour chaque pause-déjeuner $l \in L$, la pause doit débuter dans la fenêtre de temps $[e^{lunch}, l^{lunch}]$. La durée d'une pause-déjeuner est de s^{lunch}. La fenêtre de temps et la durée des pauses-déjeuner sont communes à toutes les pauses-déjeuner.

La pause-déjeuner est modélisée par un sommet et est planifiée sur un arc reliant une livraison à une collecte. La pause-déjeuner peut se situer sur le point de livraison (après le service de livraison) ou sur le point de collecte (avant le service de collecte).

En se basant sur cette modélisation, pour chaque tournée k, nous appelons A_k^{lunch}, l'ensemble des arcs du graphe où la pause-déjeuner est possible et A_k l'ensemble des arcs de la tournée. Nous avons $A^{lunch} = \bigcup_{k \in K} A_k^{lunch}$ et $A_k^{lunch} \subset A_k \subset A$. La pause-déjeuner de chaque tournée k a lieu sur un des arcs de l'ensemble A^{lunch}.

Pour un véhicule $k \in K$ donné, les arcs A_k^{lunch} où la pause-déjeuner peut avoir lieu sont donnés par l'expression suivante :

$$A_k^{lunch} = \{(i,j) \in V^2 | x_{ij}^k = 1 \wedge h_j \in [e^{lunch}, l^{lunch}] \wedge F_j \geq s^{lunch}\} \quad \forall k \in K \tag{8.18}$$

Dans l'ensemble A_k^{lunch}, nous considérons deux types d'arcs, situés directement avant l'arc de la pause-déjeuner :

- Full Edge (FE) : cet arc relie un déplacement d'un point de collecte vers un point de livraison. Durant ce déplacement, le camion est chargé (*full*) de matériaux. La pause-déjeuner aura lieu après cet arc et nous considérons la pause-déjeuner sur le point de livraison après le service de la livraison.

- Empty Edge (EE) : cet arc concerne le déplacement d'un point de livraison vers un point de collecte. Durant ce déplacement, le camion est vide (*empty*). La pause-déjeuner aura lieu après cet arc et nous considérons la pause-déjeuner sur le point de collecte avant la collecte. Le chauffeur peut attendre avant de faire la pause-déjeuner.

La figure 8.2 représente l'insertion de la pause-déjeuner après un arc FE. La pause-déjeuner a lieu au point de livraison. La figure 8.3 représente l'insertion de la pause-déjeuner après un arc EE. La pause-déjeuner a lieu sur le point de collecte. Dans les deux figures, la pause-déjeuner est représentée par une barre hachurée.

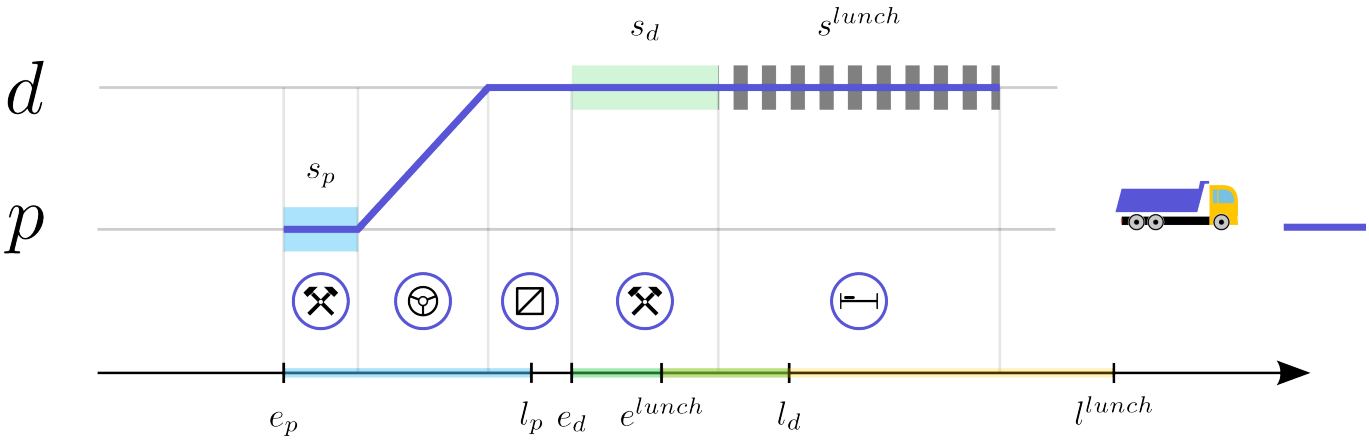


FIGURE 8.2 – Insertion d'une pause-déjeuner après un arc de type FE.

Résolution de manière exacte du problème de l'intégration des pauses-déjeuner

L'ensemble des arcs FE est noté $A_k^{lunch^F} \subset A_k^{lunch}$, et l'ensemble des arcs EE est noté $A_k^{lunch^E} \subset A_k^{lunch}$. La variable h_k^{lunch} est une variable continue qui représente l'heure de début de la pause-déjeuner du véhicule k,

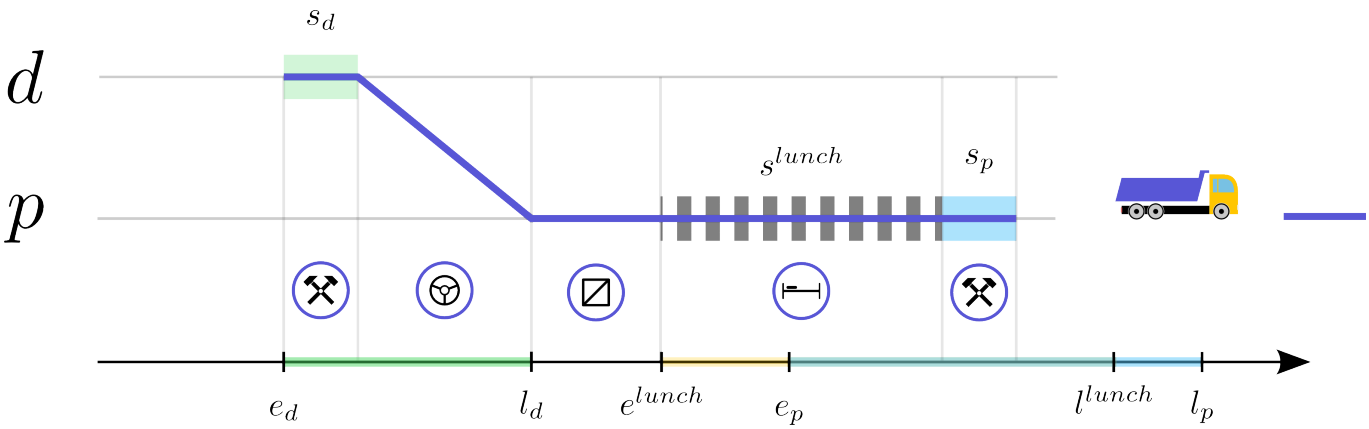


FIGURE 8.3 – Insertion d'une pause-déjeuner après un arc de type EE.

si elle est prise. La variable δ_{ij} pour $(i,j) \in A^{lunch}$ est égale à 1 si la pause-déjeuner se situe sur l'arc (i,j), 0 sinon.

Il est important de noter que ce modèle planifie la pause sur les arcs en considérant un ensemble de tournées existantes. La formulation exacte du problème d'insertion des pauses-déjeuner est la suivante :

$$\min \ \alpha * \sum_{k \in K} (h_{o'(k)} - h_{o(k)}) + \beta * C_{max} + \gamma * \sum_{k \in K} h_{o'(k)} \tag{8.19}$$

$s.t.$

$$a_i \leq h_i \qquad \forall i \in V \tag{8.20}$$

$$h_i \leq C_{max} \qquad \forall i \in V \tag{8.21}$$

$$h_i + s_i \leq h_j \qquad \forall (i,j) \in A_\Pi \tag{8.22}$$

$$h_i + t_{ij} + s_i = a_j \qquad \forall (i,j) \in A_K \backslash A^{lunch} \tag{8.23}$$

$$\sum_{(i,j) \in A_k^{lunch}} \delta_{ij} = 1 \qquad \forall k \in K \tag{8.24}$$

$$\delta_{ij} = 1 \Rightarrow h_j \geq h_k^{lunch} + t^{lunch} \qquad \forall k \in K, \forall (i,j) \in A_k^{lunch^E} \tag{8.25}$$

$$\delta_{ij} = 1 \Rightarrow h_k^{lunch} \geq h_j + s_j \qquad \forall k \in K, \forall (i,j) \in A_k^{lunch^F} \tag{8.26}$$

$$h_j + s_j + t_{jj'} + \delta_{ij} * t^{lunch} = a_{j'} \qquad \forall k \in K, \forall (i,j) \in A_k^{lunch^F}, \forall (j,j') \in A_k \tag{8.27}$$

$$a_i, C_{max} \in \mathbb{R}^+ \qquad \forall i \in V \tag{8.28}$$

$$h_i \in [e_i, h_i] \qquad \forall i \in V \tag{8.29}$$

$$h_k^{lunch} \in [e^{lunch}, l^{lunch}] \qquad \forall k \in K \tag{8.30}$$

La fonction objectif (8.19) est la même que le problème de minimisation de la durée des tournées. Les contraintes (8.20) (8.21) (8.22) sont les mêmes que les contraintes (8.7) (8.8) (8.10) du modèle $\mathcal{P}_{MINDUR}$. Les contraintes (8.23) lient l'heure d'arrivée et l'heure de service entre deux sommets d'un arc différent de celui de la pause-déjeuner. Les contraintes (8.24) imposent qu'il y ait une pause-déjeuner pour chaque véhicule. Les contraintes (8.25) lient les variables temporelles pour une pause-déjeuner après un arc FE. Les contraintes (8.26) et (8.27) lient les variables temporelles pour une pause-déjeuner après un arc FF.

En linéarisant les contraintes (8.25) et (8.26), nous pouvons résoudre ce modèle avec un solveur de PL. Cet algorithme exact d'intégration des pauses-déjeuner a été testé dans notre algorithme à diverses étapes de la recherche de solutions de l'ALNS. Pour les instances réelles de l'entreprise, il n'a pas été en mesure de retourner de solution réalisable.

Nous avons décidé de proposer une méthode heuristique pour l'intégration des temps de pauses-déjeuner.

8.2.3 Méthode heuristique pour l'intégration des pauses-déjeuner

Pour intégrer les pauses-déjeuner à une solution du problème, nous proposons une méthode heuristique. Cette heuristique comporte plusieurs fonctions appelées à différents endroits de l'algorithme de résolution.

Notre algorithme peut s'exécuter *avec* ou *sans* la prise en compte des pauses-déjeuner en fonction du type de résolution souhaitée par le décideur. Les fonctions détaillées, par la suite, sont actives lorsque la résolution du problème est effectuée avec les pauses-déjeuner.

Insertion de pauses-déjeuner

Nous proposons d'intégrer des pauses-déjeuner dès la construction initiale du planning. La méthode addLunchBreak permet de réaliser cette fonction. Avant la partie d'insertions des requêtes, nous intégrons, dans chaque tournée vide, une pause-déjeuner entre le dépôt de départ et le dépôt d'arrivée. Cette pause est géographiquement située sur le dépôt de départ.

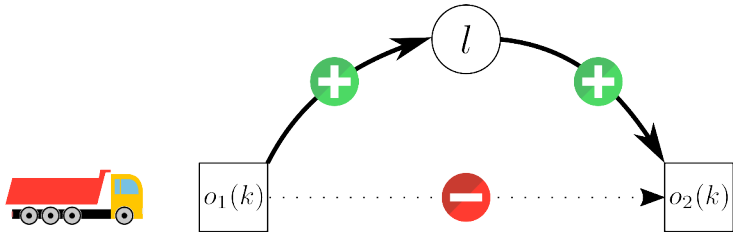


FIGURE 8.4 – Insertion d'une pause-déjeuner dans une tournée vide.

Suppression de pauses-déjeuner

À la fin de l'algorithme, il se peut que des pauses-déjeuner soient planifiées sans toutefois présenter d'intérêt. Concrètement, il s'agit des pauses-déjeuner suivantes :

- (A) tournée vide : si une tournée ne comporte aucune requête, alors nous supprimons la pause-déjeuner de cette tournée ;
- (B) tournée commençant par une pause-déjeuner : si une tournée commence par une pause, alors nous pouvons la supprimer (le véhicule ira directement servir la première requête) ;
- (C) tournée finissant par une pause-déjeuner : si une tournée finit par une pause-déjeuner, alors nous pouvons également la supprimer (le véhicule ira directement vers le dépôt d'arrivée).

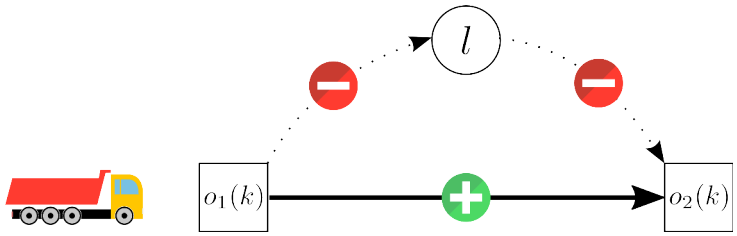


FIGURE 8.5 – Suppression d'une pause-déjeuner (A).

L'appel de la méthode removeLunchBreak remplit cette fonction en supprimant les pauses-déjeuner.

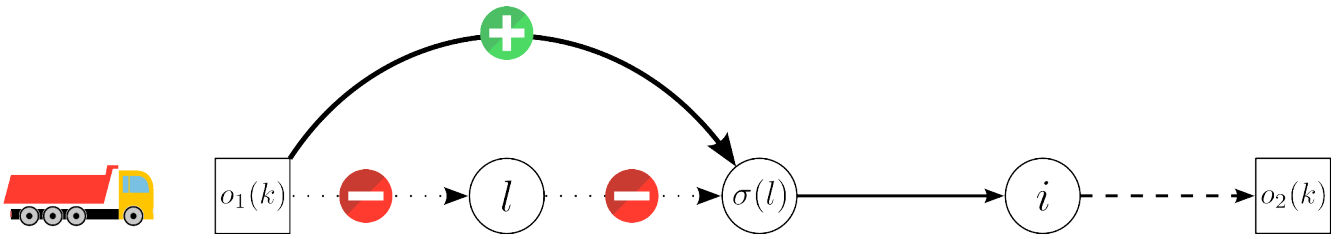


FIGURE 8.6 – Suppression d'une pause-déjeuner (B).

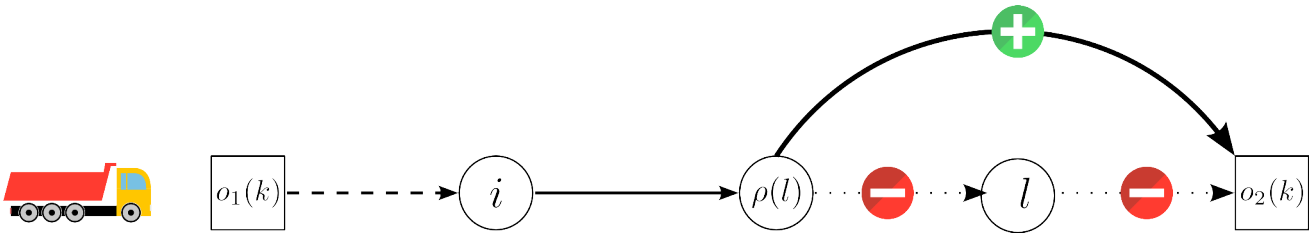


FIGURE 8.7 – Suppression d'une pause-déjeuner (C).

Déplacement des pauses-déjeuner

Lorsque les tournées comportent des pauses-déjeuner, l'appel à la méthode relocateLunchBreak permet d'optimiser le placement des pauses-déjeuner. En effet, chaque tournée comporte une pause-déjeuner. En fonction des retraits et d'ajouts de requêtes au sein de cette tournée, il peut être judicieux de relocaliser le lieu géographique de la pause-déjeuner.

Pour chaque tournée k, nous appelons l le sommet de la pause-déjeuner. Le prédécesseur de la pause-déjeuner est le sommet ρ_l et le successeur de la pause-déjeuner est le sommet σ_l. La pause-déjeuner peut avoir lieu sur le point du prédécesseur ρ_l ou sur le point du successeur. Les deux cas sont à étudier et pour chacun nous devons calculer les potentielles heures de début de la pause-déjeuner et la nouvelle heure de service du prédécesseur σ_l. Dans le cas où la pause-déjeuner aurait lieu sur le sommet ρ_l, l'heure de début de service de la pause-déjeuner est représentée par la variable h_l^ρ et l'heure de début du service du prédécesseur est noté $h_{\sigma_l}^\rho$. Dans le cas où la pause-déjeuner aurait lieu après sur le sommet σ_l, l'heure de début de service de la pause-déjeuner est représentée par la variable h_l^σ et l'heure de début du service du prédécesseur est noté $h_{\sigma_l}^\sigma$. La méthode setPoint permet de changer le lieu géographique d'un sommet du graphe G_t et la méthode getPoint permet d'obtenir le lieu géographique d'un sommet.

L'algorithme 14 présente cette méthode pour une tournée du véhicule $k \in K$.

Après chaque optimisation du placement des pauses-déjeuner, c'est à dire après chaque véhicule $k \in K$, il faut effectuer un nouvel ordonnancement au plus tôt de la solution puisque des heures de service ont pu être changées lors de l'appel de la méthode updateEarliest.

Intégration dans l'ALNS

Lorsque l'optimisation des pauses-déjeuner est activée, la méthode d'ajout addLunchBreak est appelée avant la création de la solution initiale. Elle permet d'ajouter des pauses-déjeuner dans les tournées vides de chaque camion. Pour une tournée effectuée par le véhicule $k \in K$, le point géographique de la pause-déjeuner est initialisé au dépôt de départ $o_1(k)$. La méthode de suppression removeLunchBreak est appelée à la fin de la recherche de solutions par l'algorithme ALNS.

En revanche, la méthode d'optimisation des pauses-déjeuner relocateLunchBreak est appliquée à chaque itération de l'ALNS après la phase de réparation. L'algorithme 15 propose le *framework* global de la prise en compte des pauses-déjeuner dans notre algorithme ALNS.

Pour évaluer le respect des contraintes liées aux durées de conduite journalière et continue, nous englobons ces tests dans le calcul des insertions lors de la procédure de réparation. Au prix du stockage en

Algorithme 14 : relocateLunchBreak(Route k)

début
  si ρ_l == *DEPOT* **alors**
    l.setPoint($\sigma(l)$.getPoint())
  fin
  sinon
    si ρ_l == *LIVRAISON* **alors**
      $h_l^\rho = \max\{h_{\rho_l} + s_{\rho_l}, e^{lunch}\}$
      $h_{\sigma_l}^\rho = \max\{h_l^\rho + s^{lunch} + t_{\rho_l l}^k, h_{\sigma_l}\}$
      si $h_{\sigma_l}^\rho < h_{\sigma_l} \wedge h_l^\rho \leq l^{lunch}$ **alors**
        valideAvant = VRAI
      fin
    fin
    si $\sigma(l)$ == *COLLECTE* **alors**
      $h_l^\sigma = \max\{h_{\rho_l} + s_{\rho_l} + t_{\rho_l l}^k, h_{\sigma_l}, e^{lunch}\}$
      $h_{\sigma_l}^\sigma = \max\{h_l^\rho + s^{lunch} + t_{\rho_l l}^k, h_{\sigma_l}\}$
      si $h_{\sigma_l}^\sigma < h_{\sigma_l} \wedge h_l^\sigma \leq l^{lunch}$ **alors**
        valideApres = VRAI
      fin
    fin
    si valideAvant $\wedge$ valideApres **alors**
      si $h_l^\rho < h_l^\sigma$ **alors**
        l.setPoint(ρ_l.getPoint())
      sinon
        l.setPoint(σ_l.getPoint())
      fin
    fin
    sinon si valideAvant = *VRAI* **alors**
      l.setPoint(ρ_l.getPoint())
    fin
    sinon si valideApres = *VRAI* **alors**
      l.setPoint(σ_l.getPoint())
    fin
  fin
  si *le point géographique de* l *a changé* **alors**
    updateEarliest
  fin
fin

Algorithme 15 : STRUCTURE DE L'ALNS AVEC L'INTÉGRATION DES PAUSES-DÉJEUNER

Entrées : une solution initiale $s_{initial}$,un ensemble d'opérateurs de destruction $\mathcal{N}^-$, un ensemble d'opérateurs de réparation $\mathcal{N}^+$

Sorties : **retourner** the best solution s_{best}

1 **début**
  /* Initialisation des pauses-déjeuner */
2 addLunchBreak($s_{initial}$)
3 $s_{best} \leftarrow s_{initial}$
4 $s_{current} \leftarrow s_{initial}$
5 **tant que** *critère d'arrêt non respecté* **faire**
    /* Destruction de la solution courante */
6    $s_{new} \leftarrow opDestruction(s_{current})$
    /* Réparation de la solution courante */
7    $s_{new} \leftarrow opReparation(s_{new})$
    /* Optimisation du placement des pauses-déjeuner */
8   relocateLunchBreak($s_{current}$)
    /* Acceptation de la solution courante */
9   acceptationCurrent(s_{new},$s_{current}$)
    /* Acceptation de la meilleure solution */
10   acceptationBest($s_{current}$,s_{best})
11 **fin**
  /* Suppression des pauses-déjeuner */
12 removeLunchBreak(s_{best})
13 **retourner** s_{best}
14 **fin**

mémoire des variables indiquant les temps de conduite journalier et continue, nous pouvons garder une comparaison, en temps constant, pour le respect de ces contraintes. Cette méthode de réalisabilité nous permet de maintenir nos tests d'insertion de requêtes en temps constant $\mathcal{O}_1$.

8.2.4 Expérimentations numériques

Ajout des pauses-déjeuner

Dans cette partie, nous nous intéressons à la qualité des solutions produites par l'ajout des pauses-déjeuner dans les tournées de notre problème. Nous avons testé notre algorithme ALNS en considérant une pause-déjeuner pour chaque véhicule de 45 minutes dont le début doit commencer dans l'intervalle $[12h, 13h30]$. L'horizon de temps des instances est $[6h, 20h]$. Nous utilisons les paramètres présentés dans le tableau 8.6.

Pour comparer ces résultats avec les instances industrielles, nous avons exécuté l'ALNS sur le problème FT-PDP-RS pour lequel nous avons réduit de 45 minutes l'horizon de temps pour chaque véhicule. Cette réduction de l'horizon de temps permet de nous comparer sur des instances qui sont proches, mais pas identiques.

Le tableau 8.9 présente les résultats sur les instances réelles en considérant l'ajout des pauses-déjeuner. Nous avons exécuté 10 fois l'algorithme sur chaque instance. Nous comparons le coût des solutions par rapport à la fonction objectif f définie dans l'équation 7.1.

	ALNS sans pauses-déjeuner (-45 min.)					ALNS avec pauses-déjeuner				
Instance	Best value	*moy.*	$GAP_{BestValue}$	$\#_{route}$	*moy.* w	Best value	*moy.*	$GAP_{BestValue}$	$\#_{route}$	*moy.* w
OS22	4526.55	4826.64	6.63	10.9	1001.71	4601.87	4685.92	1.83	9.6	410.93
OS32	7638.23	7971.53	4.36	16.9	1902.41	7347.2	7598.43	3.42	14.6	879.62
OS49	10441.8	10669.02	2.18	20.8	2625.46	9664.03	10120.94	4.73	19.3	1665.68
OU35	4263.14	4471.77	4.89	11.6	919.89	4009.93	4193.33	4.57	10.3	78
OU51_1	7308.17	7462.04	2.11	16.5	1462.69	6850.67	6988.78	2.02	16.2	389.78
OU51_2	6767.69	7040.01	4.02	16.1	1209.52	6494.54	6722.43	3.51	14.9	362.29
OU85	8018.21	8307.44	3.61	17.6	1045.17	7824.08	7995.15	2.19	16.8	471.37
moy.			3.97					3.18		

Tableau 8.9 – Ce tableau présente les résultats de l'ALNS sans ou avec la considération des pauses-déjeuner. Pour chaque résolution, nous présentons la meilleure solution obtenue Best Value, la moyenne des solutions, la déviation moyenne à la meilleure solution, la moyenne du nombre de tournées et la moyenne des temps d'attente.

Nous remarquons deux comportements de notre algorithme sur les instances industrielles. Premièrement, les solutions de l'ALNS avec la prise en compte des pauses-déjeuner présentent de meilleurs coûts (fonction objectif f) et des solutions avec moins de temps d'attente. Cette première analyse confirme les conclusions énoncées dans la section 8.1.5. Deuxièmement, les solutions comportent moins de véhicules en moyenne, environ 1 véhicule de moins par instance.

Il faut nuancer ces analyses, relativement bonnes, par le fait que nous ne résolvons pas exactement les mêmes problèmes. Toutefois, notre méthode est pertinente pour résoudre une instance industrielle avec la prise en compte des pauses-déjeuner.

Ajout des pauses-déjeuner et des durées maximales de conduite

Dans les tests suivants, nous reprenons l'algorithme précédent prenant en considération les pauses-déjeuner. A celui-ci, nous ajoutons la prise en compte de la durée maximale de conduite journalière (9h) et la durée maximale de conduite continue (4h30). Ces dernières contraintes sont évaluées en temps constant lors du test de réalisabilité d'une insertion.

Nous comparons les résultats de l'ALNS avec les pauses-déjeuner et l'ALNS comportant les pauses-déjeuner et la durée maximale de conduite. Cette dernière configuration correspond au problème Rich-FT-PDP-RS. Les résultats sont présentés dans le tableau 8.10. Nous avons effectué 10 exécutions de l'algorithme sur chaque instance. Nous présentons le coût de la fonction objectif f ainsi que la moyenne des temps d'attente.

	ALNS avec pauses-déjeuner					ALNS avec pauses-déjeuner et durées temps de conduite				
Instance	Best value	*moy.*	$GAP_{BestValue}$	$\#_{route}$	*moy.* w	Best value	*moy.*	$GAP_{BestValue}$	$\#_{route}$	*moy.* w
OS22	4601.87	4685.92	1.83	9.6	410.93	4373.98	4608.28	5.36	10.9	397.71
OS32	7347.2	7598.43	3.42	14.6	879.62	7269.89	7533.37	3.62	16.9	918.27
OS49	9664.03	10120.94	4.73	19.3	1665.68	9723.26	10023.06	3.08	20.8	1692.46
OU35	4009.93	4193.33	4.57	10.3	78	4024.35	4199.97	4.36	11.6	99.85
OU51_1	6850.67	6988.78	2.02	16.2	389.78	6761.2	6992.36	3.42	16.5	418.95
OU51_2	6494.54	6722.43	3.51	14.9	362.29	6480.94	6683.74	3.13	16.1	401.89
OU85	7824.08	7995.15	2.19	16.8	471.37	7832.09	8062.96	2.95	17.6	484.55
moy.			3.18					3.70		

Tableau 8.10 – Ce tableau présente les résultats de l'ALNS avec la considération des pauses-déjeuner et de l'ALNS résolvant le Rich-FT-PDP-RS. Pour chaque résolution, nous présentons la meilleure solution obtenue Best Value, la moyenne des solutions, la déviation moyenne à la meilleure solution, la moyenne du nombre de tournées et la moyenne des temps d'attente.

Ces résultats montrent que l'ajout des contraintes liées à la durée maximale des temps de conduite n'augmente pas la coût de la fonction objectif f. Nous obtenons des résultats légèrement inférieurs et tout en conservant une stabilité de notre algorithme. En revanche, l'ajout des contraintes liées à la durée des temps de conduite provoque une augmentation du nombre de véhicules utilisés dans les solutions. Les temps d'attente augmentent légèrement mais pas de manière significative.

8.2.5 Conclusion

Nous avons étudié la prise en compte des pauses-déjeuner et de durée maximale de conduite pour le problème FT-PDP-RS.

Dans un premier temps, nous avons proposé une formulation exacte du problème d'insertions des pauses-déjeuner dans notre algorithme. Cette formulation n'est pas résolue par un solveur. Nous avons donc proposé une méthode heuristique pour intégrer les pauses-déjeuner dans les tournées des camions.

La gestion des pauses-déjeuner dans notre algorithme fournit des solutions très réalistes en terme de coût en comparaison des solutions pour lesquelles nous avons diminué l'horizon de temps. Nous notons même une diminution des temps d'attente.

Ensuite, nous avons résolu le problème Rich-FT-PDP-RS sur les instances industrielles en ajoutant la prise en compte de temps maximum de conduite. Les solutions fournies restent dans les mêmes ordres de grandeur que les solutions précédentes (avec uniquement l'ajout des pauses-déjeuner). En revanche, la prise en compte de ces contraintes augmente le nombre de véhicules utilisés dans les solutions.

9

Conclusion

Le problème de collectes et livraisons en camions pleins avec contraintes de synchronisation aux ressources a fait l'objet de peu d'attention dans la littérature. Nous avons présenté une méthode d'optimisation permettant de résoudre un cas industriel de ce problème. Successivement, nous avons résolu une version simplifiée du problème avant d'ajouter progressivement des contraintes fortes jusqu'à résoudre le problème complet. Cette thèse est une contribution pour résoudre ce type de problème et apporte une lecture sur des points difficiles de la thématique des synchronisations dans les problèmes de tournées de véhicules. Elle permet à un décideur logistique d'avoir une connaissance des résultats sur les différents problèmes souhaités et de connaître la qualité des solutions pouvant être espérées. Dans la suite, nous présentons des conclusions sur nos travaux et nous proposons des perspectives de travail.

Contributions

Tout d'abord, cette thèse s'est déroulée dans un contexte industriel en partenariat avec une entreprise de travaux publics. Ces travaux représentent une véritable innovation dans le secteur des travaux publics et constituent une première avancée dans la résolution des problèmes en camions complets avec synchronisation aux ressources. La démarche de collaboration entre le monde académique et le monde industriel a permis d'ouvrir et d'enrichir un débat inédit. L'enjeu de ces travaux de thèse est de valider les propositions scientifiques sur un échantillon de données issu de cas d'études réels. Ce fût effectué par la réalisation de tests sur des instances réelles de l'entreprise. Le faible volume de données présent, pour l'instant, est dû à la jeunesse des recherches sur cette thématique. Les délais imposés dans le cadre de cette collaboration ont permis de parcourir un ensemble de thématiques tout en gardant beaucoup de problématiques ouvertes. Cela ne fait qu'encourager les futures recherches.

Nous avons proposé une modélisation du problème complet industriel Rich Full Truckload Pickup and Delivery Problem with Resource Synchronization. Ce problème n'a pas été résolu de manière exacte car les temps de calcul, sur un problème plus simplifié, ne correspondent pas aux standards de l'industrie. Nous avons donc procédé en différentes étapes pour résoudre le problème complet. Tout d'abord, nous avons proposé des modélisations différentes pour le problème de découpage des demandes en requêtes. Ces modélisations permettent de prendre en compte les différentes situations industrielles de l'entreprise : cas monopériodique ou cas multipériodique. Ces travaux constituent la première phase de notre algorithme.

Ensuite, nous avons proposé une métaheuristique ALNS pour résoudre le problème de création de tournées. Tout d'abord, nous avons proposé une formulation d'une version simplifiée du problème où nous considérons uniquement la collecte et la livraison en camions complets avec des contraintes de synchro-

nisation aux ressources. Ce problème, appelé Full Truckload Pickup and Delivery Problem with Resource Synchronization (FT-PDP-RS), a été résolu avec succès sur différentes instances. Sur des instances de la littérature, l'ALNS a fourni des solutions d'aussi bonne et de meilleure qualité. Sur des instances industrielles, l'ALNS s'est montré très compétitif en terme de temps de calcul et de qualité de solution par rapport à un solveur exact. Ces différents tests ont permis de montrer la stabilité et la qualité de notre métaheuristique ALNS. Une modélisation des contraintes de synchronisation sur des ressources a été proposée et permet de modéliser un grand nombre de cas industriels. Aussi, des avancées scientifiques ont été amenées sur les tests de réalisabilité d'une solution. Une nouvelle méthode en temps constant est proposée pour tester la réalisabilité d'insertion de requêtes dans les tournées. Au bénéfice du temps de calcul, elle permet de parcourir un plus grand espace de recherche de solutions.

La mutualisation des demandes au sein d'une même tournée permet d'améliorer légèrement la phase de découpage. En revanche, nous avons montré que cette mutualisation des demandes lors du découpage n'apportait pas d'intérêt majeur durant la deuxième phase de création des tournées.

Dans un second temps, nous nous sommes intéressés à la minimisation de la durée des tournées. Nous avons proposé des algorithmes et des opérateurs de réparation pour évaluer le coût lié à l'augmentation de la durée des tournées d'une insertion. Ces contributions innovantes permettent de résoudre le problème en minimisant les temps d'attente dans les solutions par rapport à celles retournées précédemment. Nous remarquons une nette amélioration des solutions : les tournées sont légèrement plus coûteuses en coût de trajet (+0.5%) mais comportent beaucoup moins de temps d'attente (-7.7%). Ces résultats confortent le choix des méthodes développées.

Enfin, nous avons résolu le problème complet Rich-FT-PDP-RS en incluant, d'abord la prise en compte des pauses-déjeuner, ensuite, le respect de temps de conduite maximum des tournées. Les solutions retournées conservent une bonne qualité (peu d'augmentation des coûts kilométriques et horaires). En revanche, le nombre de véhicules augmente, ce qui est une conséquence logique de la diminution du temps alloué à chaque véhicule.

Tous ces résultats ont montré une grande stabilité de notre algorithme pour résoudre différentes versions du problème industriel de collectes et livraisons en camions complets, avec contraintes de synchronisation aux ressources, pauses-déjeuner et durée maximale de conduite. Les solutions fournies sont de bonne qualité et garantissent des temps d'attente maîtrisés dans chacune des résolutions.

Perspectives

Le cadre de réalisation de cette thèse a offert de bons résultats pour le problème industriel. En revanche, certains axes de recherche ont uniquement été évoqués sans être abordés scientifiquement. Plusieurs perspectives de travail sont proposées pour continuer le travail de cette thèse.

Premièrement, la résolution de manière exacte des problèmes de tournées de véhicules en camions complets avec contraintes de synchronisation reste un défi pour proposer des solutions dans des temps honorables de calcul. Les méthodes heuristiques restent de bons candidats alliant bonne qualité de la solution finale et temps corrects de résolution. Les réelles difficultés de notre problème sont le découpage des requêtes et la synchronisation des tournées. La même analyse a été faite dans les travaux de Schmid [94]. Une méthode exacte à base de génération de colonnes pourrait être une piste de réflexion pour résoudre des problèmes modestes et surement suffisant pour certaines entreprises de travaux publics.

En ajoutant de nouvelles bornes sur la première phase de découpage et en considérant une fonction objectif incluant des coûts supplémentaires, nous serions en mesure de fournir plus d'informations pour la construction des tournées. Ce découpage pourrait être remis en cause à l'issu de la phase de création des tournées en analysant les requêtes qui n'ont pas été insérées.

La principale piste de recherche selon moi est l'ordonnancement des requêtes sur les ressources. De nombreux algorithmes existent pour fournir de très bonnes tournées aux problèmes de type PDPTW. En revanche, avec les contraintes de synchronisation aux ressources, nous avons remarqué qu'il est difficile

de trouver un bon ordonnancement des requêtes. En considérant un ensemble de tournées fixées, nous pouvons nous poser la question de réordonnancer les requêtes sur les ressources dans l'objectif de proposer un planning avec moins de temps d'attente par exemple.

Également, les solutions retournées par l'algorithme peuvent souffrir de manque de robustesse. Le décalage temporel d'une requête a une incidence sur une grande partie des tournées. Il est intéressant d'étudier la production de solutions plus robustes limitant cet impact temporel. Cette perspective rejoint l'aspect dynamique qui n'est pas abordé dans cette thèse. Il sera judicieux de pouvoir inclure des demandes, ou des requêtes, de manière dynamique au sein d'une solution existante. Cet aspect permettrait de considérer plus facilement des demandes de l'entreprise arrivant au cours d'une journée à cause d'événements propres liés à l'activité de l'entreprise. Des demandes ou requêtes peuvent être alors déplacées d'une période aux périodes adjacentes.

En perspective du travail réalisé, il pourra être traité de manière approfondie la législation sur les chauffeurs en incluant le fractionnement de la pause-déjeuner, la durée maximale de travail hebdomadaire. Sans utiliser un algorithme dédié, il apparaît judicieux d'inclure ces contraintes légales dans les tests de réalisabilité des insertions. Dans la même veine de recherche, nous avons débuté des travaux sur la minimisation de durée de tournées synchronisées. Le calcul de l'augmentation réelle de la durée des tournées lors d'une insertion représente un véritable challenge scientifique pour les heuristiques à base d'insertions.

Liste des acronymes

2E-MTVRP-SS Two-Echelon Multiple-Trip Vehicle Routing Problem with Satellite Synchronization. 47

AGV Automated Guided Vehicles. 58

ALNS Adaptive Large Neighborhood Search. 39, 47, 57, 58, 60, 91, 123

ARP Arc Routing Problem. 37

CB Camion-Benne. 28

DARP Dial A Ride Problem. 46

DARPT Dial-A-Ride Problem with Transfers. 46, 58

DVRP Dynamic Vehicle Routing Problem. 61

FSM Fleet Size and Mix problem. 44, 45

FT-PDP-RS Full Truckload Pickup and Delivery Problem with Resource Synchronization. 7, 49, 88–90, 94, 103, 104, 119, 122, 123, 126, 131, 137, 145, 146, 148

FTPDPTW Full Truckloads Pickup and Delivery Problem with Time Windows. 44, 45, 80

GES Gaz à Effet de Serre. 9–11

HFS Hybrid Flow Shop. 129

HVRP Heterogeneous Vehicle Routing Problem. 44, 45

HVRPFD Heterogeneous VRP with Fixed Costs and Vehicle-Dependent Routing Costs. 45

LNS Large Neighborhood Search. 39, 46, 90, 91, 93, 137

LTSP Log Truck Scheduling Problem. 41, 42, 107, 114

MCNF Minimum Cost Flow Problem. 40, 41, 63

MDCARPFL Multi-Depot Capacitated Arc Routing Problem with Full Truckloads. 44

MDVSP Multiple-Depot Vehicle Scheduling Problem. 43

MILP Mixed Integer Linear Programming. 58

MIP Mixed Integer Programming. 40, 41, 58, 60, 107, 119

PDP Pickup and Delivery Problem. 37, 38

PDPT Pickup and Delivery Problem with Transfers. 7, 89, 103

PDPTW Pickup and Delivery Problem with Time Windows. 34, 37–39, 57, 90, 148

PL Programme Linéaire. 59, 131, 140

PLNE Programmation Linéaire en Nombre Entiers. 44, 137

R&R Ruin and Recreate. 91

Rich-FT-PDP-RS Rich Full Truckload Pickup and Delivery Problem with Resource Synchronization. 85, 90, 146–148, 158

SA Simulated Annealing. 39, 91, 93

SANRP Synchronized Arc and Node Routing Problem. 47

SDVRP Split Delivery Vehicle Routing Problem. 15, 45

SIG Système d'Information Géographique. 11

SR Semi-Remorques. 28

TA Treshold Accepting. 91

TP Transportation Problem. 63, 72

TRM Transport Routier de Marchandises. 9, 10, 153

TSP Traveling Salesman Problem. 10, 37, 45, 47

TSPTW Traveling Salesman Problem with Time Windows. 7, 89, 103

TTVRP Timber Transport Vehicle Routing Problem. 42

VLNS Variable Large Neighborhood Search. 41, 58

VNS Variable Neighborhood Search. 40, 41, 47

VRP Vehicle Routing Problem. 14, 37, 40, 44, 45, 47

VRPCD Vehicle Routing Problem with Cross Docking. 47

VRPDWH Vehicle Routing Problem with Drivers' Workign Hours. 137

VRPFL Vehicle Routing Problem with Full Truckloads. 43, 44

VRPPD Vehicle Routing Problem with Pickup and Delivery. 38

VRPTW Vehicle Routing Problem with Time Windows. 38, 101, 137

VRPTWDR Vehicle Routing Problem with Time Windows with Drivers Rules. 137

VRSP-TW Vehicle Routing and Scheduling Problem with Time Windows. 47

Liste des figures

Liste des exemples

Liste des tableaux

Glossaire

Demande Une demande est un ordre de transport d'un matériau entre un point de collecte et un point de livraison. La quantité de matériau nécessitant d'être transporté excède généralement la capacité des camions. Une demande est constituée de plusieurs requêtes à l'issu de la phase 1.

Finisseur Un finisseur est l'outil qu'utilise les camions pour appliquer les enrobés sur une chaussée.

Flux Dans notre problème, nous distinguons deux flux de matériaux : (i) les flux tendus (pour les demandes à flux tendus) et (ii) les flux détendus (pour les demandes à flux détendus). Pour les flux détendus, chaque requête doit être livrée dans la fenêtre de temps de la demande. Les flux détendus modélisent l'approvisionnement en graviers d'un chantier ou bien encore le transfert de matériaux entre plateformes. Pour les flux tendus, nous imposons la non discontinuité dans le service. Ainsi, chaque requête doit être immédiatement suivie d'une autre requête, et ce, sans coupure. La fenêtre de temps des requêtes d'une demande à flux tendus est modifiée pour permettre la non discontinuité. Les flux tendus modélisent l'application des enrobés.

Graphe temporel Le graphe temporel est un graphe dont les sommets sont les requêtes (un sommet pour la collecte et un sommet pour la livraison), les dépôts de chaque véhicule. Nous le qualifions de temporel puisqu'il modélise également les précédences entre chaque sommet. Dans ce graphe, nous calculons les ordonnancement au plus tôt ou optimal représentant ainsi les heures de passage à chaque point donc chaque collecte ou livraison.

Opération Une opération est l'action effectuée par un camion sur un site de collecte ou de déchargement. Les transports étant en camions pleins, une opération sur un site de collecte est un chargement du camion et l'opération sur un site de livraison est un déchargement du camion.

Requête Une requête est la livraison d'une fraction d'une demande par un camion. C'est le produit du découpage d'une demande par la phase 1 de l'algorithme.

Ressource Une ressource symbolise un service ou un outil disponible sur un site de l'entreprise. Cette ressource peut-être un pont-bascule, un finisseur ou encore une pelle-mécanique. Dans notre problème, à chaque instant, un seul camion peut utiliser la ressource. Nous dirons que les ressources sont de capacité unitaire.

Rotation Une rotation est la trajet aller-retour qu'effectue un camion pour servir une fraction d'une demande. Elle correspond à une pratique métier où le responsable logistique définit un ensemble de rotations pour satisfaire les demandes.

Site Un site représente un lieu géographique où s'effectue des activités de l'entreprise. Les sites concernés dans cette thèse sont : les dépôts, les plateformes de stockage, les plateformes de recyclage, les carrières, les centrales d'enrobés et les décharges.

Synchronisation Le terme synchronisation définit l'interdépendance qui existe entre deux camions. Les ressources sont les lieux où s'effectuent la synchronisation entre deux camions.

Temps d'attente Le temps d'attente w_i correspond au temps qu'attend un camion avant d'effectuer son service au sommet i. Il est égal à la différence de l'heure de début de service moins l'heure d'arrivée $w_i = h_i - a_i$.

Temps de service Le temps de service est le temps que passe un camion pour effectuer son service. Il existe deux temps de service : (i) les temps de service sur les routes et (ii) les temps de service sur les ressources. Le temps de service sur la route est égale au temps total que prend le camion pour effectuer son opération (chargement ou déchargement, pesée sur le pont-bascule). Le temps de service sur une ressource est le temps que le camion utilise la ressource. Il peut l'utiliser uniquement pour se peser (pont-bascule) ou bien totalement (finisseur).

Temps de trajet Le temps de trajet correspond aux temps que met un camion à aller d'un point vers un autre. Le temps de trajet dépend de la nature du camion.

Bibliographie

[1] S. Afifi, D.-C. Dang, and A. Moukrim. Heuristic solutions for the vehicle routing problem with time windows and synchronized visits. *Optimization Letters*, 2015. 47

[2] D. Applegate and W. Cook. A computational study of the job-shop scheduling problem. *ORSA Journal on computing*, 3(2) :149–156, 1991. 90

[3] C. Archetti and M. G. Speranza. Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, 19 :3–22, 2012. 15, 45

[4] C. Archetti, M. G. Speranza, and a. Hertz. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40(1) :64–73, feb 2006. 45

[5] C. Archetti and M. Savelsbergh. The trip scheduling problem. *Transportation Science*, 43(4) :417–431, nov 2009. 137

[6] C. Archetti and M. G. Speranza. *The Vehicle Routing Problem : latest advances and new Ccallenges*, chapter The Split Delivery Vehicle Routing Problem : a survey, pages 103–122. Springer US, 2008. 15

[7] S. Arunapuram, K. Mathur, and D. Solow. Vehicle routing and scheduling with full truckloads. *Transportation Science*, 37(2) :170–182, may 2003. 43, 44, 49, 50

[8] L. Asbach, U. Dorndorf, and E. Pesch. Analysis, modeling and solution of the concrete delivery problem. *European Journal of Operational Research*, 193(3) :820–835, mar 2009. 41, 49, 50, 58, 72

[9] R. Baldacci, E. Bartolini, and A. Mingozzi. An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, 59(2) :414–426, 2011. 38

[10] R. Baldacci, M. Battarra, and D. Vigo. Routing a heterogeneous fleet of vehicles. In B. Golden, S. Raghavan, and E. A. Wasil, editors, *The Vehicle Routing Problem : Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 3–27. Springer, 2008. 45

[11] M. Ball, B. Golden, A. Assad, and L. Bodin. Planning for truck fleet size in the presence of a common-carrier option. *Decision Sciences*, 14(1) :103–120, 1983. 43

[12] M. Battarra, J.-F. Cordeau, and M. Iori. *Pickup-and-Delivery Problems for goods transportation*, chapter 6, pages 161–191. Society for Industrial and Applied Mathematics, 2014. 38, 40

[13] R. Bent and P. V. Hentenryck. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, 33(4) :875 – 893, 2006. 39

[14] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems : a classification scheme and survey. *Top*, 15(1) :1–31, apr 2007. 38

[15] L. Bodin, B. Golden, A. Assad, and Ball. Routing and scheduling of vehicles and crews : the state of the art. *Computers & Operations Research*, 10(2) :63–211, 1983. 43

[16] A. Bortfeldt, T. Hahn, D. Männel, and L. Mönch. Hybrid algorithms for the vehicle routing problem with clustered backhauls and 3d loading constraints. *European Journal of Operational Research*, 243 :82–96, 2015. 111

[17] M. Brachner. Solution methods for combined scheduling and transportation problems. Master's thesis, Høgskolen i Molde - Vitenskapelig høgskole i logistikk, 2013. 40, 49, 50

[18] D. Bredström, M. Rönnqvist, and D. Bredstrom. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1) :19–31, nov 2008. 47, 107

[19] P. Brucker. *Scheduling algorithms*. Springer, 2006. 101, 129

[20] A. Caris and G. Janssens. A local search heuristic for the pre- and end-haulage of intermodal container terminals. *Computers & Operations Research*, 36(10) :2763–2772, oct 2009. 45

[21] C. Chengbin and M.-C. Portmann. Scheduling to minimize total waiting time. *Applied Stochastic Models and Data Analysis*, 9 :177–185, 1993. 129

[22] G. U. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4) :568–582, 1964. 44, 45

[23] J.-F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8) :928–936, aug 2001. 42

[24] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, and M. M. Solomon. The VRP with time windows. *Les cahiers du GERAD*, 99, 2000. 34

[25] J.-F. Cordeau, G. Laporte, and A. Mercier. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, 55(5) :542–546, may 2004. 47, 103

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009. 106

[27] R. H. Currie and S. Salhi. Exact and heuristic methods for a full-load, multi-terminal, vehicle scheduling problem with backhauling and time windows. *Journal of the Operational Research Society*, 54(4) :390–400, 2003. 44, 49, 50

[28] R. H. Currie and S. Salhi. A tabu search heuristic for a full-load, multi-terminal, vehicle scheduling problem with backhauling and time windows. *Journal of Mathematical Modelling and Algorithms*, 3(3) :225–243, 2004. 44, 49, 50

[29] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4) :393–410, 1954. 10, 37

[30] E. Demir, T. Bektaş, and G. Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2) :346–359, dec 2012. 110, 111

[31] U. Derigs and T. Döhmer. Indirect search for the vehicle routing problem with pickup and delivery and time windows. *OR Spectrum*, 30(1) :149–165, 2008. 129

[32] G. Desaulniers, J. Desrosiers, A. Erdmann, M. M. Solomon, and F. Soumis. The vehicle routing problem. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter VRP with Pickup and Delivery, pages 225–242. Society for Industrial and Applied Mathematics, 2001. 38

[33] J. Desrosiers, G. Laporte, M. Sauve, F. Soumis, and S. Taillefer. Vehicle routing with full loads. *Computers & Operations Research*, 15 :219–226, 1988. 15, 43

[34] K. F. Doerner and J.-J. Salazar-González. *Pickup-and-Delivery Problems for people transportation*, chapter 7, pages 193–212. Society for Industrial and Applied Mathematics, 2014. 38

[35] M. Drexl. Synchronization in vehicle routing - a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46(3) :297–316, mar 2012. 15, 29, 46, 47, 58, 129

[36] M. Dror and P. Trudeau. Savings by split delivery routing. *Transportation Science*, pages 141–145, 1988. 15, 45

[37] G. Dueck and T. Scheuer. Threshold accepting : A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 175 :161–175, 1990. 91

[38] Y. Dumas, J. Desrosiers, and F. Soumis. The Pickup and Delivery Problem with Time Windows. *European Journal of Operational Research*, 54(1) :7–22, 1991. 34, 38

[39] M. T. Durbin. *The dance of the thirty-ton trucks : dispatching and scheduling in a dynamic environment*. PhD thesis, George Mason University, 2003. 40, 49, 50

[40] M. T. Durbin and K. Hoffman. The dance of the thirty-ton trucks : dispatching and scheduling in a dynamic environment. *Operations Research*, 56(1) :3–19, 2008. 41, 49, 50

[41] M. Ebben, M. van der Heijden, and a. van Harten. Dynamic transport scheduling under multiple resource constraints. *European Journal of Operational Research*, 167(2) :320–335, dec 2005. 58, 101

[42] M. Ehrgott. *Multicriteria Optimization*. Springer Science & Business Media, 2006. 131

[43] N. El Hachemi, I. El Hallaoui, M. Gendreau, and L.-M. Rousseau. Flow-based integer linear programs to solve the weekly log-truck scheduling problem. *Annals of Operations Research*, 232(1) :87–97, 2015. 43, 49, 50

[44] N. El Hachemi, M. Gendreau, and L.-M. Rousseau. A heuristic to solve the synchronized log-truck scheduling problem. *Computers & Operations Research*, 40(3) :666–673, feb 2013. 42, 49, 50, 107

[45] P. Flisberg, B. Lidén, and M. Rönnqvist. A hybrid method based on linear programming and tabu search for routing of logging trucks. *Computers & Operations Research*, 36(4) :1122–1144, apr 2009. 42, 49, 50

[46] Fédération Nationale des Travaux Publics. Rapport d'activité. http://www.fntp.fr/upload/docs/application/pdf/2015-01/fntp_ra_2013_web.pdf, 2013. 9

[47] M. Gendreau, G. Laporte, C. Musaraganyi, and É. D. Taillard. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 26(12) :1153–1173, 1999. 45

[48] A. Goel. Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, 43(1) :17–26, feb 2009. 137

[49] A. Goel and V. Gruhn. Drivers' working hours in vehicle routing and scheduling. In *IEEE ITSC*, pages 1280–1285, 2006. 137, 138

[50] A. Goel and L. Kok. Efficient scheduling of team truck drivers in the european union. *Flexible Services and Manufacturing Journal*, 24(1) :81–96, 2011. 137

[51] B. Golden, A. Assad, L. Levy, and F. Gheysens. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1) :49–66, 1984. 44

[52] P. Grangier, M. Gendreau, F. Lehuédé, and L.-M. Rousseau. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. Technical Report July, CIRRELT-2014-33, 2014. 47, 48, 57, 110

[53] P. Grangier, M. Gendreau, F. Lehuédé, and L.-M. Rousseau. A large neighborhood search based matheuristic search for the vehicle routing problem with cross-docking. To appear, 2015. 47

[54] M. Gronalt, R. F. Hartl, and M. Reimann. New savings based algorithms for time constrained pickup and delivery of full truckloads. *European Journal of Operational Research*, 151(3) :520–535, dec 2003. 44, 49, 50

[55] M. Gronalt and P. Hirsch. Log-truck scheduling with a tabu search strategy. In K. Doerner, M. Gendreau, P. Greistorfer, W. Gutjahr, R. Hartl, and M. Reimann, editors, *Metaheuristics*, pages 65–88. Springer US, 2007. 42, 49, 50

[56] C. Hempsch and S. Irnich. Vehicle routing problems with inter-tour resource constraints. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem : Latest Advances and New Challenges SE - 19*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 421–444. Springer US, 2008. 46

[57] P. Hirsch. Minimizing empty truck loads in round timber transport with tabu search strategies. *International Journal of Information Systems and Supply Chain Management*, 4(2) :15–41, jan 2011. 42, 49, 50, 106, 107, 109, 114, 115, 116

[58] F. L. Hitchcock. The distribution of a product from several sources to numerous localities. *J. Math. phys*, 20(2) :224–230, 1941. 63

[59] A. Imai, E. Nishimura, and J. Current. A lagrangian relaxation-based heuristic for the vehicle routing with full container load. *European Journal of Operational Research*, 176(1) :87–105, jan 2007. 46

[60] S. Irnich. A unified modeling and solution framework for vehicle routing and local search-based metaheuristics. *INFORMS Journal on Computing*, 20(2) :270–287, 2008. 16

[61] G. Kindervater and M. Savelsbergh. Vehicle routing : Handling edge exchanges. *Local Search in Combinatorial Optimization*, pages 337–360, 1997. 91

[62] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598) :671, 1983. 91, 93

[63] R. Kolisch. Serial and parallel resource-constrained project scheduling methods revisited : Theory and computation. *European Journal of Operational Research*, 90(2) :320–333, apr 1996. 101

[64] F. Lehuédé, R. Masson, S. N. Parragh, O. Péton, and F. Tricoire. A multi-criteria large neighbourhood search for the transportation of disabled people. *Journal of the Operational Research Society*, 65(7) :983–1000, may 2013. 110

[65] H. Li and A. Lim. A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, 12(02) :173–186, 2003. 39

[66] X. Li, S. C. Leung, and P. Tian. A multistart adaptive memory-based tabu search algorithm for the heterogeneous fixed fleet open vehicle routing problem. *Expert Systems with Applications*, 39(1) :365–374, 2012. 45

[67] R. Liu, Z. Jiang, R. Y. K. Fung, F. Chen, and X. Liu. Two-phase heuristic algorithms for full truckloads multi-depot capacitated vehicle routing problem in carrier collaboration. *Computers & Operations Research*, 37(5) :950–959, may 2010. 44, 49, 50

[68] Z. Liu, M. Li, and Y. Zhang. A simulation model to analyse conjoint rules for production scheduling and vehicle dispatching in rmc plants. In *Asian Conference of Management Science & Applications*, Sanya (China), 2011. 41

[69] Z. Liu, Y. Zhang, and M. Li. Integrated scheduling of ready-mixed concrete production and delivery. *Automation in Construction*, 48 :31–43, dec 2014. 41, 49, 50

[70] Q. Lu and M. M. Dessouky. A new insertion-based construction heuristic for solving the pickup and delivery problem with hard time windows. *European Journal of Operational Research*, 175 :672–687, 2005. 39

[71] R. Masson, F. Lehuédé, and O. Péton. Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, 41(3) :211–215, may 2013. 103, 104, 133

[72] R. Masson, F. Lehuédé, and O. Péton. The dial-a-ride problem with transfers. *Computers & Operations Research*, 41 :12–23, jan 2014. 46, 48, 57, 58, 110, 111

[73] Y. Mati, S. Dauzère-Pérès, and C. Lahlou. A general approach for optimizing regular criteria in the job-shop scheduling problem. *European Journal of Operational Research*, 212(1) :33–42, 2011. 129

[74] Ministère de l'environnement, de l'énergie et de la mer - Commissariat général au Développement durable. Chiffres clés du transport - Édition 2016. http://www.statistiques.developpement-durable.gouv.fr/fileadmin/documents/Produits_editoriaux/Publications/Reperes/2016/reperes-transport-ed2016-2.pdf, 2016. 9

[75] S. Mitrovic-Minic. Pickup and delivery problem with time windows : A survey. *SFU CMPT TR*, 12 :1–43, 1998. 38

[76] W. P. Nanry and J. W. Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B : Methodological*, 34(2) :107 – 121, 2000. 39

[77] M. Palmgren, M. Rönnqvist, and P. Värbrand. A column generation algorithm for the log truck scheduling problem. Technical report, Department of Science and Technology, LiTH-MAT-R, Linkoping University, Norrkoping Sweden, 2001. 42, 49, 50

[78] M. Palmgren, M. Rönnqvist, and P. Värbrand. A near-exact method for solving the log-truck scheduling problem. *International Transactions in Operational Research*, 11 :447–464, 2004. 42, 49, 50

[79] Parlement européen et Conseil européen. Regulation (ec) 561/2006, 2006. http://eur-lex.europa.eu/legal-content/FR/TXT/HTML/?uri=CELEX:32006R0561&from=FR. 137, 138

[80] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2) :81–117, 2008. 38

[81] A. Pessoa, M. Poggi de Aragão, and E. Uchoa. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. In C. Demetrescu, editor, *Experimental Algorithms*, volume 4525 of *Lecture Notes in Computer Science*, pages 150–160. Springer Berlin Heidelberg, 2007. 45

[82] V. Pillac. *Dynamic vehicle routing : solution methods and computational tools*. PhD thesis, École des Mines de Nantes, France, 2012. 61

[83] E. Prescott-Gagnon, G. Desaulniers, M. Drexl, and L.-M. Rousseau. European driver rules in vehicle routing with time windows. *Transportation Science*, 44(4) :455–473, nov 2010. 137

[84] C. Prins. Efficient heuristics for the heterogeneous fleet multitrip vrp with application to a large-scale real case. *Journal of Mathematical Modelling and Algorithms*, 1 :135–150, 2002. 45

[85] S. Ropke and J.-F. Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3) :267–286, 2009. 38

[86] S. Ropke, J.-F. Cordeau, and G. Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4) :258–272, 2007. 38

[87] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4) :455–472, 2006. 39, 57, 91, 93, 94, 103, 110, 114

[88] A. Rubasheuskaya. Combined scheduling-transportation model. Veidekke industri as case study. Master's thesis, Høgskolen i Molde - Vitenskapelig høgskole i logistikk, 2012. 40, 49, 50

[89] R. Ruiz and J. A. Vázquez-Rodríguez. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1) :1–18, 2010. 129

[90] M. Rönnqvist, S. D'Amours, A. Weintraub, A. Jofre, E. Gunn, R. Haight, D. Martell, A. Murray, and C. Romero. Operations research challenges in forestry : 33 open problems. *Annals of Operations Research*, 232(1) :11–40, 2015. 43

[91] M. A. Salazar-Aguilar, A. Langevin, and G. Laporte. The synchronized arc and node routing problem : application to road marking. *Computers & Operations Research*, jan 2013. 47, 49, 50

[92] M. W. P. Savelsbergh. Local search in routing problems with time windows. *Operations Research*, 4 :285–305, 1985. 47, 103

[93] M. W. P. Savelsbergh. The vehicle routing problem with time windows minimizing route duration. *OSRA Journal on Computing*, 4 :146–154, 1992. 47, 103, 126, 128, 132, 133

[94] V. Schmid. *Trucks in movement : hybridization of exact approaches and variable neighborhood search for the delivery of ready-mixed concrete*. PhD thesis, University of Vienna, 2007. 40, 49, 50, 148

[95] V. Schmid, K. F. Doerner, R. F. Hartl, and J.-J. Salazar-González. Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. *Computers & Operations Research*, 37(3) :559–574, mar 2010. 40, 49, 50, 58, 72

[96] V. Schmid, K. F. Doerner, R. F. Hartl, M. W. P. Savelsbergh, and W. Stoecher. A hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, 43(1) :70–85, feb 2009. 41, 49, 50, 58, 72

[97] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2) :139–171, apr 2000. 91

[98] P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, Department of Computer Science, University of Strathclyde, Scotland, 1997. 97

[99] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher and J.-F. Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin Heidelberg, 1998. 39, 90, 91, 97

[100] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2) :254–265, 1987. 101, 137

[101] É. D. Taillard. A heuristic column generation method for the heterogeneous fleet vrp. *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, 33(1) :1–14, 1999. 45

[102] P. Toth and D. Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2002. 37, 88

[103] P. Venkateshan and K. Mathur. An efficient column-generation-based algorithm for solving a pickup-and-delivery problem. *Computers & Operations Research*, 38(12) :1647–1655, dec 2011. 44, 49, 50

[104] A. Weintraub, R. Epstein, R. Morales, J. Seron, and P. Traverso. A truck scheduling system improves efficiency in the forest industries. *Interfaces*, 26(4) :1–12, 1996. 42, 49, 50

UNIVERSITE BRETAGNE LOIRE

MINES Nantes
Institut Mines-Télécom

Thèse de Doctorat

Axel GRIMAULT

Optimisation de tournées de camions complets dans le secteur des travaux publics

A pickup and delivery problem with full truckloads in the public works sector

Résumé
Le transport de matériaux pour la réalisation d'infrastructures routières et le terrassement représente, en 2013, plus de la moitié de l'activité du secteur des travaux publics. Les méthodes d'optimisation de tournées de véhicules permettent aujourd'hui de résoudre des problèmes de grandes tailles en intégrant les contraintes liées au métier. Dans cette thèse, nous nous intéressons à la résolution du problème riche de collectes et livraisons en camions complets avec des contraintes de synchronisation sur les ressources. Dans un premier temps, nous résolvons le problème de tournées de véhicules avec une méthode heuristique en deux phases. Dans un second temps, nous étudions l'intégration des contraintes liées aux temps de conduite des chauffeurs ainsi que l'ajout des pauses déjeuners aux tournées. Nous testons les algorithmes proposés sur des instances de la littérature et des instances réelles issues d'une application industrielle d'une entreprise de Travaux Publics.

Mots clés
recherche opérationnelle, problème de collectes et livraisons, camions complets, synchronisation, métaheuristique.

Abstract
In 2013, the transportation of materials for roads construction and earthwork represents more than half of the whole activity of in the public works sector. Optimization methods for vehicle routing problems allow to solve big-size problems with industrial sector constraints. In this thesis, we focus on solving the rich full truckload pickup and delivery problem with resource synchronization. First, we solve this vehicle routing problem with a two phase heuristic method. Then, we study the integration of regulation of drivers' working hours and the addition of lunch breaks in routes of vehicles. These methods are tested on instances from the literature and real life instances from a public works company.

Key Words
operations reseach, pickup and delivery problem, full truckloads, synchronization, metaheuristic.